Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

Vyhodnocení a srovnání autentizačních metod v Low-Code platformě Mendix

DIPLOMOVÁ PRÁCE

Studijní program: Informační management

Autor: Tomáš Matějka, Bc.

Vedoucí diplomové práce: prof. Ing. Petr Doucek, CSc.

Praha, květen 2023

Poděkování

Chtěl bych poděkovat především prof. Ing. Petru Douckovi, CSc. za veškerou poskytnutou odbornou pomoc a vedení. Dále poté kolegům, se kterými jsem mohl konzultovat problémy, a následně ještě všem, co mne podporovali při vypracovávání práce.

Abstrakt

Low code je mladý, inovativní, rapidně se vyvíjející a populární způsob tvorby aplikací, primárně webových. Oproti klasickým přístupům vývoje pracuje s předdefinovanými funkcemi a extensivně využívá drag & drop funkcionality, díky čemuž dokáže být rychlejší, levnější a pohodlnější, je avšak méně flexibilní.

Ačkoliv se jedná o populární a perspektivní odvětví, neexistuje mnoho literatury pojednávající o této oblasti. Proto vnikla tato diplomová práce, kde se autor zabývá jednou z nejpopulárnějších low code platforem – Mendixem. Práce se zaměřuje na oblast autentizace.

Hlavním cílem práce je vymezit, analyzovat, vyhodnotit z různých pohledů/dimenzí a následně srovnat možnosti autentizace dostupné v aktuální verzi Low-Codové platformy Mendix. Dílčími cíli práce je doporučit nejvhodnější metody autentizace pro aplikace různých velikostí, potřeb a účelů a dále vytvoření demo aplikace využívající rozebírané autentizační metody.

Těchto cílů je dosaženo pomocí analýzy jednotlivých autentizačních metod, které jsou hodnoceny ze 3 dimenzí – použitelnosti, bezpečnosti a nákladovosti (na zdroje i znalosti). Dimenze obsahují kritéria, dle kterých jsou metody hodnoceny, a na základě tohoto hodnocení je vypracován celkový přehled a doporučení pro jednotlivé aplikace. Zároveň jsou tyto metody implementovány do demo projektu.

Jako nejvhodnější byly vyhodnoceny metody autentizace pomocí jména a hesla, následně SSO řešení, kde obecně nejlepší postavení mají metody vyvíjené samotným Mendixem. Výjimkou je biometrická autentizace, která se naopak umístila v polovině výsledného žebříčku. Na základě těchto výsledků byly v rámci doporučení navrhovány primárně tyto metody, kde výjimkou bylo firemní ERP, kdy kvůli vyšším nárokům na bezpečnost je vhodné použít i více faktorovou autentizaci. Dále práce poskytuje mnoho informací o fungování jednotlivých metod.

Klíčová slova

Analýza, Autentizace, Autorizace, Low-Code, Mendix, Hodnocení autentizačních metod.

JEL klasifikace

M15 IT Management, D80 Informace, znalosti a nejistota, O33 Technologická změna.

Abstract

Low code is an innovative, rapidly evolving, and popular approach for creating applications, primarily web applications. It tries to make development faster, cheaper, and easier than traditional development by using pre-defined components and drag & drop interfaces. However, it lacks the flexibility of traditional coding.

In contrast to its popularity and promise, there is a lack of literature in this area. This thesis addresses this gap by analyzing one of the most popular low-code platforms, Mendix, with a focus on the area of authentication.

The main objective of the thesis is to evaluate the authentication options available in the current version of Mendix by defining, analyzing, and comparing them from multiple perspectives, including usability, security, and cost. The sub-objectives are to recommend the most suitable authentication methods for applications of different sizes, needs, and purposes and to create a demo application using the discussed authentication methods.

These objectives are achieved by analyzing each authentication method and evaluating it from 3 dimensions - usability, security, and cost (in both resources and knowledge). The dimensions contain criteria against which the methods are evaluated, and based on this evaluation, an overall ranking and recommendations for the applications are created. At the same time, these methods are implemented into a demo project.

Based on these findings, the thesis recommends login + password and SSO methods for most applications, with the exception of corporate ERP systems, where multi-factor authentication is also appropriate due to higher security requirements. The thesis provides valuable insights into the authentication options available in Mendix and offers recommendations for their use in various applications.

Keywords

Analysis, Authentication, Authorization, Evaluation of authentication methods, Low-Code, Mendix.

JEL Classification

M15 IT Management, D80 Information, Knowledge, and Uncertainty, O33 Technological Change.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Téma práce

Low-code je specifický způsob vývoje aplikací, který se snaží (mimo jiné) zkvalitnit, zrychlit a ulehčit vývoj aplikací. Jde o mladý způsob, který se stále rapidně vyvíjí a o kterém se stále přesně neví, jak moc se uchytí v budoucnu a co od něj očekávat (tzv. „Code vs Low-Code vs No-Code“). Přednostmi vývoje v tomto prostředí je časté používání modelů, bohaté využívání vizuálních prvků, programování pomocí tzv „Drag & Drop interface“, snadné škálování, neopakování kódu (pomocí tvorby modulů), nebo také snadná čitelnost kódu a dokumentace.

Práce se věnuje analýze a hodnocení autentizačních metod dostupných v Low-Code platformě Mendix. Tato platforma patří mezi dominantní, kde spolu s dalšími jako jsou OutSystems, Salesforce, Appian, Oracle či Power Apps tvoří hlavní jméno a podobu celého odvětví. Jedná se o původem holandskou firmu, nedávno koupenou a nově spadající pod Siemens. Aktuálně jde o leadera „magického kvadrantu“ dle firmy Gartner a jednu z nejinovativnějších platforem v odvětví (Wong et al. 2021).

Důvody zvolení tématu

Důvodů pro zvolení tématu je více:

Autentizace a autorizace jsou nedílnou součástí každé moderní aplikace. Zabezpečují správný chod a chování nejen desktopových, ale i mobilních či webových aplikací. Jejich důležitost a nutnost použití je nediskutabilní, a právě proto chci využít příležitosti se jimi zabývat při psaní této práce.

Jak již bylo dříve zmíněno, Low-code je nový způsob vývoje, kvůli čemuž se mnoho bakalářských, diplomových či jiných prací touto tématikou nezabývá, ačkoliv autor v této oblasti vidí potenciál. Proto by se chtěl specificky zaměřit na autentizaci právě v tomto prostředí.

Posledním důvodem je osobní zájem autora o tyto oblasti. V oblasti Low-Code má několik let praxe i certifikací a autentizace (spolu s oblastí kyberbezpečnosti obecně) patřila mezi nejzajímavější okruhy ze studia. Proto rád uvítá možnost psaní práce zabývající se těmito tématy.

Cíle práce

Hlavním cílem práce je vymezit, analyzovat, vyhodnotit z různých pohledů/dimenzí a následně srovnat možnosti autentizace dostupné v aktuální verzi Low-Codové platformy Mendix. Průběh výběru metod a důvody vybrání daných metod jsou vytyčeny v metodice.

Dílčími cíli práce je:

Doporučit nejvhodnější metody autentizace pro aplikace různých velikostí, potřeb a účelů.

Vytvoření demo aplikace využívající rozebírané autentizační metody. Bude se jednat o aplikaci spustitelnou na lokálním prostředí (v dále uvedené verzi Mendix Studia Pro), přičemž odkaz na aplikaci a mpk soubor[1] by byl poskytnut spolu s touto prací.

Způsob dosažení cílů

Cílů bude dosaženo několika kroky: Prvně bude proveden výběr metod, které budou dále hodnoceny. Společně s tímto krokem budou vytyčeny dimenze hodnocení, kde pro každou dimenzi bude stanoveno, jak bude hodnocena, jaké jsou možné úrovně hodnocení a jejich rozsahy, jaké faktory do dimenze spadají a proč byla tato dimenze vybrána. Jako poslední krok této přípravy budou vybrány aplikace pro tvorbu doporučení.

Následně bude každé vybrané metodě autentizace věnována kapitola, ve které bude provedeno hodnocení a bude v ní také rozebráno, jak je možné danou metodu implementovat v prostředí Mendix Studia Pro, což bude prakticky použito při tvorbě demo aplikace.

Poté dojde k celkovému srovnání metod, rozboru a interpretaci výsledků, spolu s tvorbou doporučení pro dříve vybrané aplikace.

Nakonec dojde k ověření výsledků práce. To proběhne formou dotazníků a zkrácených verzí výsledků, které budou zaslány odborníkům z praxe.

Přesné definice průběhů, výběrů, hodnocení apod. jsou uvedeny v kapitole metodika.

Definice rozmezí a limitací práce

Práce se zabývá metodami autentizace dostupnými jak ve webových Mendix aplikacích, tak i v nativních (tj. mobilních) aplikacích Mendix (tím je možné použít i mobilní funkce jako je biometrická autentizace).

[1] formát Mendix projektu

Práce bude využívat jak metody autentizace dostupné defaultně v platformě, tak i různé autentizační moduly, které je možné stáhnout z prostředí „Mendix Marketplace“.

Práce se zabývá primárně metodami autentizace dostupnými v platformě Mendix v době psaní práce (aktuálně je poslední dostupná verze 9.18.0 (k 2.10.2022)). Při vývoji aplikace bude použita poslední dostupná verze, avšak může dojít k výjimkám, kde bude nutné použít verzi starší (např. pokud by některý z modulů vyšší verzi nepodporoval).

Práce bude zahrnovat primárně uživatelskou autentizaci (tj. nezahrnuje scénáře, kdy spolu komunikuje více aplikací a potřebují se navzájem ověřit např. pro využití REST rozhraní).

Struktura práce

V kapitole Úvod a Metodika jsou popsány základní informace o práci, spolu s metodickým popisem práce.

Hlavní obsah práce je možné rozdělit do 4 hlavních částí. V první části, do které bych zahrnul kapitoly 3 a 4, je rozebíráno prostředí, ve kterém se práce pohybuje. Jde o popis autentizačních a autorizačních metod spolu s rozebráním platformy Mendix a Low-Code odvětví obecně.

V části druhé, kam bych zařadil kapitoly 5 a 6, jsou rozebírány vybrané autentizační metody dle metodiky práce a následně jsou klasifikovány dle úrovní jednotlivých dimenzí. Každé metodě je věnována vlastní podkapitola a jsou zde zmíněny i metody jako více faktorová autentizace apod.

Třetí část práce, kapitola 7, interpretuje výsledky předchozích kapitol a poskytuje ucelené a okomentované finální hodnocení. Cílem této kapitoly je přehledně znázornit výsledky hodnocení a porovnávání.

Poslední, čtvrtá část práce, složená z kapitol 8 a 9 slouží k ověření výsledků práce a poskytuje závěr celé práci. Zároveň jsou zde použity výstupy předchozích částí ke tvorbě doporučení pro vybrané aplikace.

Na konci práce je poté možno nalézt ještě seznam použité literatury a přílohy, ve kterých se nachází dotazník pro ověření výsledků práce společně s odpověďmi respondentů a výsledná demo aplikace a odkaz na ni.

1 Rešerše

1.1 Low Code

Jelikož se jedná o nové odvětví vývoje, které se stále rychle vyvíjí a formuje, nenachází se v této oblasti příliš mnoho publikací. Autoři, kteří se této oblasti věnují se často soustředí na porovnávání jednotlivých Low Code platforem navzájem či s tradičním programováním a snaží se hodnotit jejich silné a slabé stránky. Často se také rozebírá Low Code ve vztahu k moderním a inovačním technologiím, jako je například IoT, AI či machine learning a mnohdy se zabývají digitální transformací. Jako příklad je možné zde uvést (Sanchis et al. 2020), (Maiya 2022) či (Sahay et al. 2020).

Publikace bývají jak teoretické, tak i praktické, avšak málokdy zachází do technických detailů. Mnohdy se také jedná spíše o analýzy veřejného mínění či analýzy využívanosti, které jsou prováděny za pomoci dotazníkového šetření.

Velkou limitací v mnoha publikacích, je dle mého názoru nerozlišování mezi platformami, kdy poté často dochází ke různým pojetím, co vše je, a co již není Low Code. To je problematické, jelikož poté mnoho autorů vnímá jinak hranici např. mezi Low Code a No Code. Je zde problém, že neexistuje jednotná definice Low Code jako takového.

Nejkvalitnějšími publikacemi v oblasti bych označil dříve zmíněný článek (Sahay et al. 2020) a následně ještě studii firmy Gartner (Wong et al. 2021), které se obě věnují definování Low Code odvětví, platforem a zároveň dokáží vhodně představit jejich rozdíly.

Dalšími kvalitními zdroji informací v této oblasti jsou ještě dokumentace a publikace samotných Low Code firem k jejich platformách.

1.2 Autentizační a autorizační metody

Oproti publikacím v oblasti Low Code, zde existuje mnoho publikací, které komplexně pokrývají celou problematiku autentizace, autorizace a i identifikace. Pro účely práce je možné například vytyčit publikace (Boonkrong 2020) či (Wilson a Hingnikar 2019), které se zabývají identity a access managementem a všemi faktory v nich vystupujících. Pokrývají jak oblast kryptografie a samotných metod, tak i přístupy k řízení bezpečnosti a doplňují je o reálné příklady a možnosti využití v praxi, což je pro tuto práci velmi přínosné.

Je možné nalézt i publikace, které se detailně zaměřují na jednotlivé metody, jako je například publikace (Richer a Sanso 2017), ve které se autoři věnují výhradně autorizačnímu protokolu OAuth 2. Podobných publikací existuje mnoho, což je pro práci velkým přínosem.

Dále je důležité zmínit publikace OWASP[2], neziskové organizace, které se zabývají bezpečností software. Známý a vlivný je například tzv. OWASP Top 10 (OWASP 2021b), který dokonce existuje i ve verzi pro Low Code prostředí (OWASP 2022a).

Poslední publikace, které bych chtěl v této oblasti zmínit jsou bakalářské, diplomové či jiné kvalifikační práce z této oblasti. Ačkoliv nedosahují často takové odbornosti jako předchozí publikace, často je možné nalézt zajímavé pohledy a názory. Zde jako příklad uvádím (Šíma 2021) a (Kučera 2012).

1.3 Hodnocení autentizačních metod

V této oblast dle mého názoru není dostatek publikací, což ovšem vyplývá z její náročnosti. Objektivně zhodnotit jednotlivé autentizační metody je obtížné a náročné, přičemž mnohokrát může mnohem více záležet na implementaci než na samotné metodě.

Nejvhodnější a nejkvalitnější publikace z této oblasti je článek (Helkala a Snekkenes 2009), ve kterém se autoři snaží detailně formalizovat jednotný proces hodnocení autentizačních produktů, který by se zabýval jak bezpečnostní stránkou, tak použitelností, nákladovou stránkou či kompatibilitou se zařízeními a v určité míře i uživateli. Problémem této hodnotící metody pro účely této práce ovšem je, že pracuje s přesnými daty, jako například cena nutného hardware či administrační náklady, pravděpodobnosti krádeže či entropií hesel.

(Khan 2017) je další přínosnou publikací, která se zabývá vhodností autentizačních metod pro chytré telefony. Zde je definován vlastní systém hodnocení a porovnávání metod, který je ovšem mnohem méně důkladný než předchozí příklad, ale zato je použitelnější v širší míře.

V neposlední řadě bych chtěl ještě zmínit bezpečnostní dokumenty a doporučení neziskových či vládních organizací, jako je např (NISO 2005), (NIST 2020) či (MVČR 2020), které doporučují či přímo stanovují určité úrovně bezpečnosti a způsoby jejich určení. Pro účely práce jsou tyto dokumenty velmi přínosné.

[2] Open Web Application Security Project

2 Metodika

2.1 Výběr autentizačních metod

Práce se zabývá všemi autentizačními metodami, které jsou v době psaní práce dostupné v platformě Mendix defaultně (Mendix 2022g), nebo které jsou veřejně dostupné ke stažení na tzv. Mendix Marketplace[3] (Mendix 2022j). Výjimkou je přihlašování pomocí OTP (One time password), pro který se mi nepodařilo najít modul, ale chtěl bych jej v práci zahrnout, a proto jsem se rozhodl vytvořit modul vlastní, za použití defaultně dostupných nástrojů. Pro účely práce jsou proto celkově zohledněny následující metody:

- Přihlášení pomocí jména a hesla
    - Spolu s rozšířením CAPTCHA
- OTP,
- SSO,
- OIDC a OAuth,
- LDAP,
- biometrické přihlášení pomocí otisku prstu a
- více faktorová autentizace pomocí:
    - emailu,
    - Google Authenticatoru.

Níže je možné nalézt přehled použitých modulů a jejich tvůrců pro každou metodu. Z funkčního hlediska jsou metody více rozebrány v rámci svých podkapitol v kapitole 5. Všechny metody je možné použít jak ve webových, tak v nativních aplikacích, kromě biometrického přihlašování, které funguje jen v nativních aplikacích.

Přihlášení pomocí jména a hesla

Modul: -

Autor: Mendix

Odkaz: -

Jedná se o defaultní autentizační metodu, kterou platforma poskytuje. Je dostupná v každém projektu od vytvoření aplikace.

[3] Jedná se o veřejný prostor platformy Mendix, kam mohou uživatelé nahrávat různé komponenty, moduly nebo doplňky a odkud si je poté mohou jiní stahovat.

CAPTCHA

Modul: Recaptcha plugin (v2.0.0)

Autor: Objectivity

Odkaz: https://marketplace.mendix.com/link/component/115762

OTP

Modul: -

Autor: Autor Práce

Odkaz: -

Jednoduchý modul zajišťující autentizaci, nebo sloužící jako další faktor při přihlášení. V době psaní práce žádný oficiální nebo komunitní modul na Mendix marketplace není dostupný, proto bude během psaní práce doplněn autorem.

SSO

Modul: MendixSSO (v3.1.1)

Autor: Mendix

Odkaz: https://marketplace.mendix.com/link/component/111349

OIDC a OAuth

Modul: OIDC SSO (v2.0.0)

Autor: Mendix

Odkaz: https://marketplace.mendix.com/link/component/120371

LDAP

Modul: LDAP Synchronization module (v8.0.0)

Autor: Mendix

Odkaz: https://marketplace.mendix.com/link/component/24

Biometrické přihlášení

Modul: Native Mobile Resources (v3.11.1)

Autor: Mendix

Odkaz: https://marketplace.mendix.com/link/component/109513

Více faktorová autentizace (email)

Modul: Multi-factor authentication for Mendix (v2.0.0)

Autor: Appronto

Odkaz: https://marketplace.mendix.com/link/component/116892

Více faktorová autentizace (Google Authenticator)

Modul: Google Authenticator Connector

Autor: Mendix

Odkaz: https://marketplace.mendix.com/link/component/2948

2.2 Hodnocení autentizačních metod

2.2.1 Výběr dimenzí

Při hodnocení metod je použit multidimenzionální pohled, kde každá metoda je hodnocena dle předem vybraných dimenzí. Tyto dimenze byly vybrány tak, aby společně pokrývaly všechny důležité oblasti a zároveň aby se nepřekrývaly, za účelem zajištění co možná nejvíce objektivní hodnocení.

Výběr dimenzí probíhal rešerší aktuálně dostupných způsobů na hodnocení autentizačních metod, spolu s články zabývající se touto oblastí, ze kterých byly autorem konsolidovány tři finální dimenze – jedná se o použitelnost, bezpečnost a dimenze nákladů a znalostí. Popis jednotlivých dimenzí se nachází v následující kapitole 2.2.2.

Jako nejvlivnější publikace v této oblasti bych označil (Helkala a Snekkenes 2009), (NISO 2005) a (Khan 2017). Primárně prvně zmíněná publikace je pro tuto oblast velmi zajímavá, jelikož během rešerše jsem nenarazil na jinou publikaci, která by k problematice hodnocení přistoupila tak detailně a pokusila se vytvořit jednotný vzorec z tak obtížně kvantifikovatelných veličin.

2.2.2 Dimenze

Použitelnost

Tato dimenze pokrývá všechny praktické vlastnosti autentizační metody z pohledu uživatelů Patří do ní faktory jako:

- složitost pro uživatele,

- možnost využití na různých zařízeních,
- počet nutných akcí, které musí uživatel pro autentizaci vykonat a doba, než je uživatel autentizován,
- uživatelská přívětivost,
- dostupnost a
- spolehlivost.

Uživatel se bude chtít přihlásit rychle, jednoduše, pokud možno na jedno kliknutí a nebude chtít být přesměrováván nebo vyplňovat stejné údaje vícekrát. Pokud by nucen byl, je možné že začne hledat jiné služby, které naplní jeho požadavky jinde a budou pro používání snazší a přehlednější. Proto je tato dimenze při hodnocení autentizačních metod důležitá, jelikož pokud si chtějí webové aplikace či stránky udržet konkurenceschopnost a návštěvnost, musí používat takové metody, které budou ochotni používat i uživatelé.

Do této dimenze zároveň spadá i možnost propojení aplikací s jinými, za účelem zajištění SSO autentizace nebo aplikací stanovené podmínky, jako heslová politika či časové vypršení relace.

Bezpečnost

Tato dimenze zahrnuje všechny bezpečnostní faktory autentizačních metod. Spadá do ní:

- bezpečnost přenosu dat,
- charakter přenášených dat,
- bezpečnost uložení dat,
- odolnost vůči útokům,
- bezpečnostní rizika,
- používané metody,
- standardizace,
- externí rizika.

Obecně jde o standardní bezpečnostní požadavky, které by byly od autentizační metody očekávatelné. Uživatelé nebudou chtít používat aplikace se špatnou pověstí bezpečnosti a která nakládá s jejich daty nezodpovědně a nezabezpečeně. Je možné zde sledovat i náchylnost sledovaných metod k prolomení v kyberprostoru nebo v obyčejném světě například pomocí shoulder surfingu, phishingu či jiných útoků.

Bezpečnost je z těchto tří dimenzí nejobtížnější pro hodnocení a srovnávání, jelikož ji lze velmi obtížně kvantifikovat a svou roli zde má mnoho faktorů. Ku příkladu může být těžké rozhodnout, zdali je bezpečnější mít data o uživatelích uložená na své databázi nebo například na databázi poskytovatele SSO.

Na druhou stranu je tato oblast dosti standardizovaná, kde jako příklad je možné uvést zahraniční publikace NIST 800-63B (NIST 2020) či guidance of National Cyber Security Centre (NCSC 2018), zabývající se bezpečností v kyberprostoru obecně, nebo v tuzemském prostředí existuje v rámci eGovernmentu například „Dokument konkretizující minimální

požadavky na kvalifikované systémy elektronické identifikace a na prostředky pro elektronickou identifikaci“ (MVČR 2020), který se v kapitole 2.3. přímo věnuje autentizaci.

Dříve zmíněná publikace NIST je pro dimenzi bezpečnosti velmi důležitá, jelikož definuje tzv. Authenticator Assurance Levels (AAL), což jsou celkem 3 úrovně bezpečnosti autentizačních prostředků, kde pro každou jsou definované podmínky dle typu autentizace, znovu ověření, odolnosti vůči různým útokům či krádežím.

Náklady a znalosti

V případě dimenze nákladů a znalostí je možné si představit:

- náklady pořízení,
- náklady provozu,
- nutný hardware,
- čas implementace,
- nutné lidské zdroje,
- infrastrukturní požadavky, nebo
- závazky vznikající z implementace.

Jedná se o faktory, které ovlivňují nejen proces samotného zavádění, ale i chodu a údržby celého autentizačního řešení, a to ať už v časových, finančních, hardwarových, softwarových či lidských veličinách. Zároveň sem spadají i závazky a závislosti na službách třetích stran, pokud je autentizační metody využívají.

Jde spíše o dimenzi, kterou by reálně zohledňoval jen poskytovatel, kterému půjde o dosažení určité rovnováhy přínosů a výdajů. Tato hladina se bude lišit dle účelu aplikace, charakteru dat (jako citlivost, množství, tajnost), charakteru uživatelů, preferencí poskytovatele a mnoha dalších faktorů.

2.2.3 Proces hodnocení

Hodnocení bude probíhat vyhodnocováním vhodnosti či kvality metody v každé z dimenzí, kdy pro každou dimenzi je autorem vytyčeno 5 úrovní. Tyto úrovně mají jasně stanoveny kritéria, které hodnocené metody musí splňovat, aby jich mohly dosáhnout. Úrovně za sebou také následují v pevně daném sledu, a aby mohla metoda získat vyšší hodnocení, musí zároveň splňovat podmínky nižší. Jestli tomu se tak nestane, bude ohodnocena poslední úrovní, u které splňuje kritéria všechna.

Jak dimenze, tak i úrovně a jejich kritéria, jsou definovány autorem práce a jejich přehled je zpracován v následujících podkapitolách.

Použitelnost

Kritéria dimenze použitelnosti byla zvolena následovně:

Možnost použití na více zařízeních

Toto kritérium hodnotí, jestli je možné metody použít na všech klasických zařízeních, které jsou uživateli využívána. V ideálním případě bude možné metodu použít kdekoliv, bez omezení vycházejících z typu a schopnostech zařízení[4], velikosti obrazovky, použitého prohlížeče[5] apod.

Toto kritérium může nabývat 3 hodnot:

- „Ano“, pokud je metoda podporována na všech typech zařízení a na všech nejpoužívanějších prohlížečích.
- „Částečně“, pokud je metoda podporována na většině zařízeních a většině hlavních prohlížečů.
- „Ne“, pokud metoda funguje jen na jednom zařízení či prohlížeči.

Počet akcí, které musí uživatel provést

V tomto kritériu je zohledňován maximální počet akcí, které musí uživatel vykonat pro úspěšné autentizování. Pro účely kritéria jsou akce počítány jako jednotlivé úkony, které musí uživatel vykonat z již otevřené login stránky. Jako akce se počítá např. vyplnění údajů, stisknutí tlačítka, otevření jiné stránky (např. email klienta), nebo použití jiného zařízení. Počítají se pouze akce při přihlášení již registrovaného a nakonfigurovaného uživatele. Výsledné hodnoty jsou rozděleny dle následujících intervalů:

- „Pár“, pokud je v (0, 2>.
- „Několik“, pokud je v (2, 4>.
- „Více“, pokud je v (4, 6>.
- „Mnoho“, pokud je v (6, ∞).

Doba potřebná k přihlášení

Zde je zohledňováno, jak dlouho musí uživatel jednotlivé akce provádět a čekat, než je úspěšně přihlášen. Je zde zohledňována průměrná doba při maximálním počtu akcí (viz minulé kritérium). Čas je počítán od začátku první akce až do zobrazení domácí stránky po přihlášení. Možné hodnoty kritéria jsou následující:

- „Krátká“, pro doby přihlášení (0, 7> sekund.
- „Střední“, pro doby přihlášení (7, 14> sekund.
- „Spíše delší“, pro doby přihlášení (14, 25> sekund.
- „Dlouhá“, pro doby přihlášení (25, ∞) sekund.

Potřeba účtu/profilu v jiné aplikaci

V tomto případě je hodnoceno, jestli je nutné mít založený účet, či profil, v jiné aplikaci, aby bylo možné se přihlásit do aktuální aplikace. Možné hodnoty jsou:

- „Ano“, pokud je nutné takový účet vlastnit.

---

[4] Pro účely práce jsou rozlišovanými typy: PC, tablety a mobilní zařízení
[5] Primárně zohledňovány Google Chrome, Firefox, Safari, MS IE a Edge

- „Ne“, pokud to nutné není.

Uživatelský komfort

Kritérium zohledňující míru spokojenosti či frustrace, když uživatel používá danou autentizační metodu. Jde o velmi subjektivní kritérium, zabývající se obtížností pochopení principu a srozumitelností metody, kde pro co nejvyšší komfort a spokojenost uživatele by měl být schopen uživatel chápat, proč je o dané údaje žádán, proč je někam přesměrováván či proč musí dané akce vykonávat. Úrovně jsou stanoveny následovně:

- „Vysoký“, pro metody, se kterými uživatel je chopen rychle, efektivně a intuitivně pracovat za vynaložení minimálního úsilí. Uživatel ví, co od metody očekávat.
- „Střední“, pokud metodu, nemožnou zařadit do žádné ze zbývajících úrovní. Jde o pozici, ve které metoda není ani příliš komfortní pro uživatele, ale ani nekomfortní.
- „Nízký“, pro metody, které pro uživatele působí zmatečně. Dochází u nich například k mnoha přesměrováním, uživatel vyvíjí větší námahu pro jejich používání, nebo je neschopen porozumět co se po něm chce.

Spolehlivost

Z pohledu spolehlivosti je zde zohledňována primárně chybovost autentizační metody. Je zde zkoumáno, jak často dochází u metody k chybám, výpadkům a obecně problémům. Žádané je pochopitelně nenarazit na problémy žádné. Úrovně jsou stanoveny následovně:

- „Vysoká“, kdy k problémům nedochází téměř nikdy. Metoda je velmi spolehlivá a netrpí žádnými nedostatky v této oblasti.
- „Střední“, kdy k problémům někdy dochází, ale metoda je stále pro svůj účel použitelná.
- „Nízká“, kdy je pro nespolehlivost metoda nepoužitelná.

Časové vypršení relace a reautentizace

Z pohledu uživatele bude co nejpříjemnější, pokud po přihlášení zůstane co nejdéle přihlášen a nebude se muset v rozumném časovém horizontu opět přihlašovat. Na druhou stranu ale zároveň nebude chtít být přihlášen napořád. Zároveň je zde avšak komplikace, že vypršení relace není vlastností autentizační metody, ale konfigurace aplikace a také nejde stanovit ideální dobu vypršení relace. (NIST 2020) stanovuje například, aby byl nucen uživatel autentizace alespoň jednou za 12 hodin a aby byl odhlášen po alespoň 30 minutách neaktivity, (OWASP 2022c) doporučuje tuto dobu stanovit jako *minimal value possible depending on the context of the Application*[6]. Proto jsou zde stanoveny úrovně kritéria následovně:

- „Individuální“, kde je možné definovat dobu vypršení relace na úrovni samotných uživatelů či skupin uživatelů (např. rolí či funkčních skupin, nebo oddělení).

[6] Jako co nejkratší smysluplnou dobu dle kontextu (účelu) aplikace, autorský překlad.

- „Globální“, kde je administrátor, nebo jiná pověřená osoba schopna nastavit dobu vypršení relace pro všechny uživatele aplikace.
- „Pevně stanovené“, pro případy, kdy není možné stanovit dobu vypršení relace, například z důvodu použití třetí strany pro autentizaci, která tuto možnost nenabízí.
- „Žádné“, pro metody, které nijak s dobou vypršení relace nepracují.

Přehled úrovní

Tabulka 1: Přehled úrovní dimenze použitelnosti (zdroj: Autor)

Úroveň	Použitelné na více zařízeních	Počet akcí	Doba k aut.	Potřeba účtu v jiné aplikaci	Uživatelský komfort	Spoleh.	Časové vypršení relace a reauten.
1	Ne	Mnoho	Dlouhá	Ano	Nízký	Nízká	Žádné
2	Ne	Více	Spíše delší	Ano	Nízký	Střední	Pevně stanovené
3	Ne	Několik	Střední	Ano	Střední	Střední	Pevně stanovené
4	Částečně	Několik	Střední	Ano	Vysoký	Vysoká	Globální
5	Ano	Pár	Krátká	Ne	Vysoký	Vysoká	Individuální

Bezpečnost

Bezpečnost a charakter dat při přenosu

V tomto kritériu je sledováno, zdali autentizační metoda komunikuje bezpečně a zdali všechna data, která se přenášejí nejsou zranitelná. Všechna data při komunikaci by měla být šifrována a měly by být použity pouze bezpečné a ověřené protokoly, např. HTTPS. Kritéria jsou stanovena následovně:

- „Vysoká“, kdy data jsou přenášena dostatečně bezpečně, dochází tedy k šifrování přímo mezi klientem a serverem takovým způsobem, že pro nezúčastněné strany nelze zjistit obsah. Jsou užívány aktuální kryptografické algoritmy.
- „Střední“, kdy jsou data částečně chráněna, ale úroveň této ochrany není plně dostačující. Může jít například o omezené hashování či nedostatečné šifrování dat, ze kterého je možné získat v určitém čase původní podobu dat. Dalším příkladem

spadajícím do této kategorie může být přílišná závislost na „Security through obscurity[7]“.
- „Nízká“, kdy přenášená data nejsou nijak zabezpečena a přenášejí se čistě jako textové řetězce či nešifrovaná data. Nejslabší způsob zabezpečení.

Bezpečnost uložení dat

Kritérium hodnotící způsob uložení dat v databázi. Ve většině případů se bude jednat o uložení jednotlivých verifikátorů, jako pin, heslo apod., avšak je zde sledováno i nakládání s tokeny a jednorázovými kódy.

Data, která je potřeba ukládat by měla být adekvátně zabezpečena šifrováním, případně pokud to je možné dalšími bezpečnostními prvky, jako je například solení či pepření[8]. Naopak data, jako jsou OTP či jednorázové tokeny by ukládána být neměla vůbec či minimálně (Babič 2020). Kritéria stanovena následovně:

- „Vysoká“, kdy s ukládanými daty je nakládáno bezpečně. Jsou uložena v dostatečně zabezpečené formě a dle možností jsou aplikovány další bezpečnostní prvky.
- „Střední“, kdy je bezpečnost uložených dat řešena okrajově či částečně. Nejsou aplikována všechna nutná či doporučovaná opatření.
- „Nízká“, kdy nejsou aplikovány žádné bezpečnostní opatření. Vše je ukládáno v originální podobě.

Zároveň je nutné zabezpečit uloženou konfiguraci celé autentizační metody, v případě Mendixu celého autentizačního modulu. V případě, že metoda žádná data neukládá, je možné ji považovat za vysoce bezpečnou.

Odolnost vůči útokům

Zde je sledována odolnost vůči různým způsobům kybernetickým útokům. Kromě konvenčním způsobům, jako může být injection, brute force či slovníkové útoky, rainbow tables, MitM, jsou zde zahrnuty odolnosti vůči sociálnímu inženýrství, jako phishingu nebo pretextingu, či dalších způsobů.

Vyšší důraz zde bude kladen na kontrolu zranitelností zmíněných v OWASP Top 10 (OWASP 2021b) z roku 2021 a OWASP Top 10 Low Code Security Risks (OWASP 2022a).

- „Celková“, kdy metoda je schopná v dostačující míře odolat všem dříve zmíněným útokům a zároveň neobsahuje žádné známé zranitelnosti.
- „Částečná“, kdy metoda nedokáže odolat jedné či vícero hrozbám a útokům.
- „Žádná“, kdy metoda nepracuje s žádnými opatřeními a ani neřeší žádné externí hrozby.

---

[7] Jedná se o způsob zabezpečení, kdy bezpečnost SW vychází z nejasnosti a nevědomosti o jeho bezpečnostních prvcích, chování apod. Může být velmi problematické, pokud použito jako primární zdroj ochrany (CWE 2022).
[8] Viz podkapitola 5.1.1.

Rizika technologií a architektury

Kritérium pokrývající bezpečnost používaných technologií a architektury celé metody. Oproti kritériím řešící samotnou odolnost implementací autentizačních metod v rámci Low Code, zde je hodnocena rizikovost vycházející z konceptuální úrovně. Nejsou zde sledovány konkrétní způsoby práce s daty, ale celková architektura. Dále se sledují technologie, které jsou nedílnou součástí metody, pokud nějaké technologie má.

- „Nízká“, kdy použité technologie a architektura nepředstavují žádné či zanedbatelné riziko pro bezpečnost metody.
- „Střední“, kdy technologie či architektura obsahují bezpečnostní vady, které představují mírné bezpečnostní riziko a zranitelnost celé metody. Tyto vady je možné ošetřit dodatečnými bezpečnostními opatřeními
- „Vysoká“, kdy použité technologie a architektura obsahují jednu, či více seriózních vad, které mohou ohrozit bezpečný chod metody. Tyto vady není možné ošetřit dodatečnými bezpečnostními opatřeními.

Standardizace

Kritérium řešící stupeň standardizace jednotlivých metod. K posouzení bude použita veřejně dostupná dokumentace, pokud možno specializovaných národních či nadnárodních organizací. Vyšší standardizace je obecně pro metodu prospěšnější. Nejen že bude metoda snadnější na správu, dokumentaci, implementaci a integraci, primárně bude využívat ověřené metody a technologie. Je pro aplikaci mnohem bezpečnější, aby byla vyvinuta za používání otestovaných open-source metod či standardizovaných protokolů. Zároveň bude pro externí aplikace mnohem důvěryhodnější (Hickey 2021). V neposlední řadě bude jednoduší udržovat bezpečnostní algoritmy a využívané knihovny aktuální.

- „Vysoká“, kdy metoda je standardizována vícero specializovanými organizacemi či skupinou organizací, zabývající se kyberbezpečností. Obecně existuje mnoho veřejných dokumentů popisující standardy, best practices, guidelines apod.
- „Střední“, kdy metoda je standardizována jednou organizací či skupinou organizací, jako může být např. tvůrce či poskytovatel.
- „Nízká“, kdy metoda není nijak a nikým standardizována.

Přehled úrovní

Tabulka 2: Přehled úrovní dimenze bezpečnosti (zdroj: Autor)

Úroveň	Bezpečnost a charakter dat při přenosu	Bezpečnost uložení dat	Odolnost vůči útokům	Rizika technologií a architektury	Stand.
1	Nízká	Nízká	Žádná	Vysoká	Nízká
2	Střední	Střední	Částečná	Střední	Nízká
3	Střední	Střední	Částečná	Střední	Střední

4	Vysoká	Vysoká	Celková	Střední	Střední
5	Vysoká	Vysoká	Celková	Nízká	Vysoká

Náklady a znalosti

Kritéria této dimenze byla celkem zvolena 3:

- dodatečné náklady provozu,
- čas implementace a nutné lidské zdroje,
- závazky vznikající z implementace.

Pozn: Původně bylo zamýšleno v rámci dimenze nákladů a znalostí sledovat celkové náklady provozu i pořizovací náklady. To se ale projevilo jako velmi problematické, jelikož tyto ceny jsou velmi závislé na mnoha vlastnostech aplikace, provozovatele i poskytovatele, které je obtížné standardizovat a objektivně zhodnotit. Zároveň se nepodařilo nalézt ani žádné případové studie sledující nákladovost pořízení a provozu autentizačních služeb.

Dodatečné náklady provozu

Pro každou metodu autentizace bude pomocí tohoto kritéria určena dodatečná finanční náročnost pro provoz. Každá Mendix aplikace musí běžet na určité úrovni Mendix Cloudu, z jehož licence plyne několik podmínek pro počet uživatelů. Tyto podmínky jsou ale univerzální pro každou aplikaci a není možné postavit metodu tak, aby je efektivně obešla.

Toto kritérium hodnotí dodatečné náklady, které je nutné vynaložit pro provoz metody. Může jít o služby externích stran či o dokoupená rozšíření aplikace. Možné úrovně kritéria jsou:

- „Žádné“, pro metody kde nejsou žádné, či pouze velmi malé až zanedbatelné náklady plynoucí z provozu metody.
- „Nízké“, kde metoda není nákladná, avšak je nutné platit určité sumy na měsíční či roční bázi. Měsíční cena na uživatele je nižší, jak samotná cena za uživatele v rámci licence Mendixu, tj. 10 euro či 12 USD[9] (Mendix 2023d).
- „Vysoké“, kdy je nutné platit vysoké částky za provoz metody. Měsíční cena za uživatele je vyšší, jak cena za uživatele Mendixu.

Čas implementace a Nutné lidské zdroje

Kritérium měřící náročnost na implementaci, přičemž se soustředí jak na časové, tak i lidské zdroje. Jelikož se práce zabývá pouze implementacemi v prostředí Mendix Studio Pro

[9] Při neuvedení ceny za uživatele se cena za uživatele určí dodatečně tak, že celková cena bude vydělena 100, což je maximální počet uživatelů při základní licenci (Mendix 2023e).

a dílčím cílem práce je i tvorba aplikace, bude kritérium vyhodnoceno autorem při implementaci samotných metod. Pro zajištění vyšší objektivity budou ovšem využity i veřejně dostupné materiály zabývající se implementací těchto metod, jako např. (LeBlanc a Messerschmidt 2016) pro OIDC či (Richer a Sanso 2017) pro OAuth 2.

Cílem kritéria je zohlednit nutné znalosti, moduly, komponenty a další prvky, které je nutné použít pro implementaci dané metody v Mendixu, případně nutných dodatečných zařízení, serverů či aplikací. Úrovně jsou stanoveny následovně:

- „Nízké“, pokud metodu je možné implementovat do 0,5 MD, teda 4 hodin.
- „Střední“, pokud metodu je možné implementovat do 1 MD, teda 8 hodin.
- „Vysoké“, pokud metodu není možné implementovat do 1 MD.

Závazky vznikající z implementace

Poslední kritérium dimenze nákladů a znalostí, které se zabývá tím, zdali implementací dané autentizační metody vznikají či nevznikají provozujícímu subjektu nějaké nové závazky vůči třetím stranám. Pokud nějaké vznikají, pak je sledováno jaké.

Závazky, které kritérium sleduje, musí být nefinančního charakteru, protože finanční jsou již zahrnuty v nákladech na pořízení či provoz. Může tedy jít například o umístění dat mimo vlastní servery, nutnost využití služeb třetích stran či další technické, infrastrukturní, komunikační či jiné povinnosti. Možné úrovně jsou následující:

- „Nevznikají“.
- „Vznikají“.

Přehled úrovní

Tabulka 3: Přehled úrovní dimenze nákladů a znalostí (zdroj: Autor)

Úroveň	Dodatečné Náklady provozu	Čas implementace a nutné lidské zdroje	Závazky vznikající z implementace
1	Vysoké	Vysoké	Vznikají
2	Nízké	Vysoké	Vznikají
3	Nízké	Střední	Vznikají
4	Žádné	Střední	Vznikají
5	Žádné	Nízké	Nevznikají

2.2.4 Konfigurace Mendix Cloudu

Hodnocení metod bude probíhat na Mendix Free Cloud Node, jehož konfigurace je dostupná níže, viz Tabulka 4.

Tabulka 4: Konfigurace Mendix Cloudu (zdroj: Autor)

Název	**Hodnota**
Název aplikace	DP_matt10
Verze aplikace	1.0.0.75
Cloud Node	Mendix Cloud EU: AWS Ireland (Free)
URL	https://dpmatt10-sandbox.mxapps.io
Mendix Verze	9.18.0.53394

Testování nativních metod proběhne na mobilním telefonu, specifikovaným níže, viz Tabulka 5, pomocí aplikace Make It Native 9 (Mendix Research & Development 2022).

Tabulka 5: Mobilní zařízení k testování (zdroj: Autor)

Název	**Hodnota**
Verze mobilní aplikace	1.2.17.141
Mobilní zařízení	Google Pixel 7
OS mobilního zařízení	Android 13

2.3 Výběr aplikací pro tvorbu doporučení

Jedním z dílčích cílů práce je tvorba doporučení pro aplikace různých typů, potřeb a účelů. Aby bylo možné tento cíl naplnit, jsou v této kapitole autorem vybrány celkem 3 aplikace, pro které budou doporučení tvořena. Snahou bylo vybrat dostatečný počet aplikací, které se liší v několika zásadních faktorech, kvůli čemuž bude pro každou nejspíše vhodný jiný způsob autentizace.

Vybranými aplikacemi jsou: interní ERP systém, o e-shop a o veřejný zpravodajský web. Nejdůležitějšími faktory u výběru jsou:

- Počet uživatelů a denní návštěvnost
    - Jedná se o denní počet uživatelů, a to jak v podobě denního maxima, tak průměrného vytížení. Zde je možné předpokládat nejvyšší návštěvnost u zpravodajského webu a nejmenší v případě ERP systému.
- Počet anonymních uživatelů
    - Vnímáno jako počet „neregistrovaných“ či „nepřihlášených“ uživatelů, kteří aplikaci přesto využívají. Nejsou zde tímto myšleni uživatelé na login stránce webu, ale např. lidé, kteří nepřihlášeni čtou poslední zprávy.
- Počet přihlášení
    - Celkový počet pokusů o přihlášení. Je možné se domnívat, že velká většina návštěvníků zpravodajského webu nebude přihlášení využívat, zatímco v e-shopu se uživatel spíše přihlásí (tj. pokud bude něco nakupovat) a v případě ERP systému bude muset být přihlášený každý.
- Charakteru uživatelů
    - Faktor zohledňující zkušenosti, vzdělání, věk, profese, zájmy, cíle, názory či další jiné charakteristiky uživatelů aplikací. Každá aplikace je stavěna pro nějaký typ tzv. „end-usera“ a to platí i pro výběr autentizačních metod, které bude možné v aplikaci využívat. Zároveň je možné do tohoto faktoru zahrnout i zařízení, které uživatel využívá k přístupu k aplikaci, kde např. v případě ERP můžeme očekávat výhradně PC, u zpravodajského portálu či e-shopu půjde spíše o mobilní zařízení.
- Citlivostí dat
    - Zde jsou zohledňován charakter dat, se kterými aplikace či uživatelé pracují a jejich důležitostí pro chod provozující společnosti. Citlivost dat proto dále ovlivňuje, jaké metody autentizace aplikace bude povolovat, aby se k datům nemohla dostat neoprávněná osoba.
- Cíle provozovatele
    - Myšleno jako dlouhodobé cíle a očekávání provozovatele od aplikace a uživatelů. Dle těchto cílů bude aplikace stavěna a provozována určitým způsobem. Např. e-shop bude mít zájem na tom, aby si uživatel založil účet, a proto musí poskytnou takové možnosti autentizace, které budou pro uživatele schůdné, zatímco ERP systém může mít nastavené vyšší úrovně zabezpečení (např. více faktorové ověření zadání kódu z emailu, či používání čipových karet), jelikož jej zaměstnanci musí používat každopádně a pro provozovatele je velmi důležité, aby se k datům dostali jen oprávnění lidé.
    - Je důležité také si uvědomit, jaké má aplikace místo v celé strategii. Pro provozovatele zpravodajského portálu může být hlavní získat co největší návštěvnost a získat co možná nejvyšší příjmy z umístěných reklam, zatímco provozovatel e-shopu bude nejen chtít návštěvnost, ale primárně co nejvyšší prodeje a v případě ERP systému jde o IS podporující hlavní činnost společnosti, který přímo žádné finance nevytváří.

2.4 Způsob ověření výsledků

Pro ověření výsledků práce budou kontaktováni odborníci z praxe, kteří mají bohaté zkušenosti jak v oblasti Low-Code, tak v oblasti autentizačních prostředků. V příloze práce je možné nalézt dotazník (viz Příloha B: Otázky), který jim byl odeslán, společně s odpověďmi jednotlivých respondentů (viz Příloha C: Odpovědi na Otázky). Celkové

shrnutí jednotlivých odpovědí se nachází v kapitole 9. Z osobních a firemních důvodů nejsou jejich jména zveřejněna a jednotliví odborníci se v práci nazývají odborník A,B,C a D.

Celkově se pro ověření podařilo získat celkem 4 odborníky, přičemž většina z nich pochází ze zahraničí, proto je dotazník a jejich odpovědi v angličtině. Obecně jde o zaměstnance velkých, mezinárodních korporací, se kterými jsem se měl možnost setkat při své praxi s Mendixem. Každému z nich budou zaslány (autorsky přeložené) výsledky práce[10] společně s dříve zmíněným dotazníkem, aby se mohli k práci vyjádřit a co nejobjektivněji ji posoudit. Jejich odpovědi budou uvedeny neupravené v příloze, a interpretovány ve celkovém shrnutí ověření výsledků práce.

[10] Jedná se o zkrácenou verzi práce, kdy jsou odebrány kapitoly 1, 3 a 4.

3 Autentizace a autorizace

3.1 Vymezení pojmů

Celá tato kapitola se zabývá problematikou autentizace, autorizace a několika dalších relevantních a blízkých pojmů, které je nutné zmínit pro správné pochopení dalších kapitol.

3.1.1 Identifikace

Tento pojem vhodně vysvětlují 2 následující citace:

Identification is the ability to identify uniquely a user of a system or an application that is running in the system (IBM 2022).

Akt nebo proces, během kterého entita předloží systému nějaký identifikátor, na jehož základě systém může rozeznat entitu a odlišit ji od jiných entit (Pavlíček 2020, s. 3).

Z obou z nich je možné vyčíst, že jde o určení a odlišení uživatele (či aplikace) od ostatních uživatelů (aplikací) v systému. Nejde ovšem o nic více, tj. nedochází zde k žádnému ověření, zdali údaje jsou platné, aktuální, či věrohodné, pouze dojde k identifikaci entity.

Druhá citace byla vybrána záměrně proto, že navíc ještě zmiňuje existenci tzv. identifikátoru. Jedná se o jakýkoliv nástroj či informaci, která dokáže osobu unikátně identifikovat v rámci systému. Může jít například o uživatelské id, email, telefonní číslo, certifikát, či přihlašovací jméno. Důležité je ještě zdůraznit, že identifikace vždy probíhá vůči systému, kterému se snažíme identifikovat jako daný uživatel. Je možné a často i běžné, že v daném systému může mít jeden uživatel více účtů. Zde jako vhodný příklad je možné uvést osobní a firemní účet v bance (Wilson a Hingnikar 2019).

Z bezpečnostních důvodů má tato oblast různé regulační opatření, ze kterých by bylo vhodné zmínit evropský eIDAS Levels of Assurance (LoA) (European Commision 2022) vycházející z (European Commision 2014), kterým se řídí i Česká Republika v oblasti identifikačních prostředků identity občana v rámci eGovernmentu (Správa základních registrů 2022), nebo americký NIST 800-63-3 (Grassi et al. 2017).

Všechny tyto opatření slouží (nejen) ke stanovení „úrovní záruky“ a ověření pravosti v oblasti identifikace. V nařízení eIDAS se jedná o tři úrovně: nízká, značná a vysoká, v NIST 800-63-3 jde o IAL1 (nejslabší úroveň), IAL2 a IAL3 (nejsilnější). Obecně u nejslabší úrovně nedochází k žádným dodatečným úkonům, zatímco u silnějších dochází k určité úrovni validace a verifikace poskytnutých informací před přiřazením identifikátoru. Může jít například o ověření emailu, telefonního čísla nebo také jména či adresy. Zároveň je kladen důraz na kvalitu dokladů (důkazů) při přiřazování, jako je např. platnost, vydavatel, proces vydání, garance správnosti, dostupnost fotografie, nebo dostupnost biometrických údajů (Pavlíček 2020).

3.1.2 Autentizace

Autentizace úzce souvisí s identifikací. Jde o ověření, jestli je uživatel skutečně osobou, za jakou se vydává. Obecně je možné říci, že pro úspěšnou autentizaci posílá uživatel systému svůj identifikátor (např. email) společně s verifikátorem (eng. authenticator), či výstupem z něj, které systém ověří a následně uživatele autentizuje. Systém tyto údaje může ověřit, jelikož je původně získal v průběhu registrace, nebo využije službu, u které je již uživatel registrován a je schopná jej ověřit (princip SSO) (NIST 2020).

Verifikátory

Obecně verifikátor vzniká během registračního procesu, ale záleží na charakteru samotného verifikátoru, způsobu a formy registrace či na systému. Tímto pojmem označujeme velké množství různých prostředků, které slouží k autentizaci, a v následujících odstavcích bys se proto chtěl věnovat jejich charakteristice.

Základním dělením, se kterým je možné se setkat je dle podstaty (či charakteru) samotného verifikátoru. Může se totiž jednat o něco, co uživatel zná, má, nebo je (Wilson a Hingnikar 2019). Příklady uvedeny níže:

- Něco, co znám – Hesla, PINy, záložní kódy, odpovědi na bezpečnostní otázky.
- Něco, co mám – Generátor jednorázových hesel, certifikát, spárované mobilní aplikace.
- Něco, co jsem – biometrie, např. otisk prstu

Je možné setkat se i s dalšími kategoriemi tohoto dělení, ačkoliv nejsou tak časté. (Boonkrong 2020) popisuje ještě další dvě, a to:

- Něco, co zpracuji (eng. Something you process) – funguje například na principu OTP, které je tvořeno za pomoci uživatelovy tajné informace. Nikdy se ovšem neposílá celá tajná informace či její část, ale nějaký výsledek operací, domluvených při registraci. Uživatelova tajná informace může být např matice čísel, nad kterou poté uživatel provádí operace a pro autentizaci zasílá pouze výsledné číslice. Nebo naopak může jít o operace, které uživatel používá k výpočtům nad vstupními daty, které získá od systému.
- Někdo, koho znám – osoba, která je schopná autentizovat jinou osobu. Použitelné hlavně při krizových situací, kde uživatel např. zapomene heslo, a proto kontaktuje jinou osobu, která jej autentizuje vůči systému (nebo mu například vydá jiné, dočasné heslo).

Dalším dělením, které je více specifické a účelné, je dělení dle typu verifikátoru. Toto dělení je převzato z (NIST 2020):

- Memorized Secrets – Například heslo, PIN
- Look-Up Secrets – Záložní kódy, nebo také tzv. “Recovery keys”
- Out-of-Band Devices – Naskenování zaslaného QR kódu mobilem či potvrzení akce v mobilní aplikaci
- Single-Factor OTP Device – Aplikace v mobilu, generující jednorázová hesla

- Multi-Factor OTP Device – Stejné jako dřívější, ale s dodatečným faktorem autentizace, například otiskem prstu
- Single-Factor Cryptographic Software – Kryptografický klíč, nacházející se někde v systému
- Single-Factor Cryptographic Device – Kryptografické operace pomocí externího hardwarového zařízení
- Multi-Factor Cryptographic Software – Stejné jako dřívější, ale s dodatečným faktorem autentizace, například otiskem prstu
- Multi-Factor Cryptographic Devices – Opět přidán další faktor

Oproti minulému dělení, zde jsou verifikátory rozděleny do skupin dle podoby autentizačního procesu. Zároveň je možné si všimnout, že seshora dolů nabývají verifikátory na komplexnosti a zároveň i bezpečnosti. Je to, protože v publikaci jsou tyto typy použity jako jeden z požadavků pro AAL (Authentication Assurance Levels, viz 2.2.2 dimenze bezpečnosti). Ukázka těchto požadavků je zobrazena v Tabulka 6.

Tabulka 6: AAL požadavky (NIST 2020, tabulka 4–1)

Requirement	AAL1	AAL2	AAL3
Permitted authenticator types	Memorized Secret; Look-up Secret; Out-of-Band; SF OTP Device; MF OTP Device; SF Crypto Software; SF Crypto Device; MF Crypto Software; MF Crypto Device	MF OTP Device; MF Crypto Software; MF Crypto Device; or Memorized Secret plus: • Look-up Secret • Out-of-Band • SF OTP Device • SF Crypto Software • SF Crypto Device	MF Crypto Device; SF Crypto Device plus Memorized Secret; SF OTP Device plus MF Crypto Device or Software; SF OTP Device plus SF Crypto Software plus Memorized Secret

Více faktorová autentizace

Jako poslední z části o autentizaci bych se chtěl zabývat více faktorovou autentizací. Tento koncept, známý také jako vícefázová či vícekroková autentizace[11], spočívá v použití více autentizačních faktorů pro úspěšnou autentizaci. V praxi může jít například o použití jména a hesla společně s jednorázovým kódem, který přijde uživateli na email. Jde o dodatečnou úroveň ochrany, kdy útočníkovi nestačí zmocnit se pouze jména a hesla, ale i celé emailové schránky. Jedná se o lehký, ačkoliv účinný přístup k zvýšení bezpečnosti (Microsoft 2022e).

3.1.3 Autorizace

Poté, co systém identifikoval a autentizoval uživatele, musí většinou dojít k jeho autorizaci. Tímto pojmem se označuje udělení práv na určité akce, které může uživatel v rámci systému provést. K autorizaci může docházet při registraci, nebo po vstupu uživatele do systému, kdy mu jsou dříve zmíněná práva udělena dle systémového nastavení či pravidel, které jsou definovány v rámci tzv. Identity and Access Management (IAM) (Okta 2022b).

Oproti autentizaci se jedná o neviditelný process pro uživatele, který zároveň nemá ani možnost tyto práva měnit (Andrioaie 2022). Je ovšem neméně důležitý pro správný a bezpečný chod systému, kde nedostatky v této oblasti mohou vést k vyšším rizikům úniku dat, špatné správě uživatelských práv, špatné UX[12], nižší produktivitě či vyšším IT nákladům (Roy 2021).

Existuje mnoho přístupů, strategií, či modelů autorizace. Mezi nejznámější patří Role-Based Access Control a Attribute-Based Access Control, jako další je ale možné zmínit Relationship based Access Control nebo Graph-based Access Control či mnoho dalších. Pro tuto práci je nejdůležitější Role-Based Access Control, tzv RBAC, jelikož to je právě ten model, který je používán v rámci platformy Mendix. RBAC funguje na základě definování rolí. V Mendixu je možné deklarovat 1-n rolí a každé z nich specifikovat práva na zobrazování stránek, používání logických prvků, jako jsou tzv. Microflows, Nanoflows apod. a zároveň je možné definovat práva na úrovni dat, kde je možné u každé role stanovit, jestli má právo zobrazit či zapisovat některý z atributů, nebo jestli má právo vytvořit či smazat celou entitu[13].

Tyto role fungují aditivně, kdy uživateli je přiřazena minimálně jedna a maximálně neomezeně rolí, jejichž práva se sčítají, díky čemuž je možné snadno a přehledně distribuovat práva mezi uživatele. Každá implementace RBAC může být ovšem odlišná, např. může být povolena jen jedna role na uživatele, nebo může role být definovány hierarchicky (Pavlíček 2020).

---

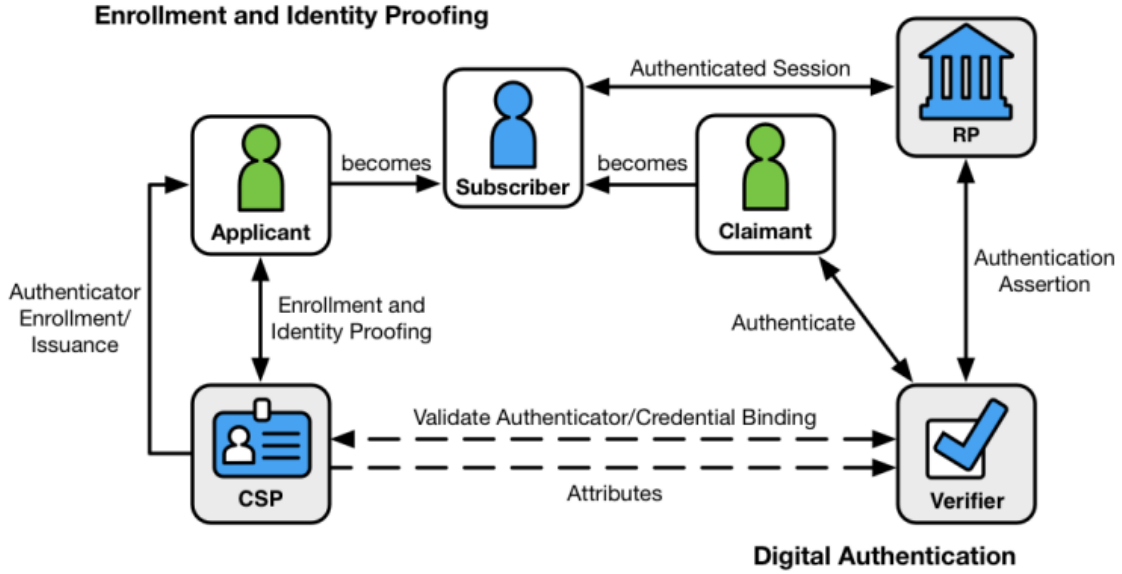
[11] Existují články a publikace, jako například (Savvy Security 2021) nebo (ISDecisions 2021), které chápou tyto pojmy rozdílně: někdy jako rozdílné podkategorie více faktorové autentizace, někdy jako jiné prvky. (Pavlíček 2020, s. 24) naopak uvádí: *v praxi nemá smysl rozlišovat pojmy 2FA a 2SV*. V práci nejsou pojmy rozlišovány.

[12] User Experience

[13] Pro vysvětlení pojmů viz kapitola 4.2

3.2 Využití

Identifikace, autentizace a autorizace dohromady tvoří prostředky, kterými je možné ověřovat totožnost a určovat oprávnění uživatelů. Jak již bylo dříve nastíněno, existuje mnoho způsobů autentizace, autorizace i identifikace, stejně tak jako typů verifikátorů a způsobů celkové integrace. Základní principy bývají avšak obvykle stejné a vhodně je popisuje NIST ve své publikaci 800-63-3 (Grassi et al. 2017). V této publikaci je představen tzv. Digital Identity Model, který je znázorněn na Obrázek 1 níže a který se snaží představit jednotlivé základní aktéry a interakce při tvorbě digitální identity.



Obrázek 1: NIST Digital Identity Model (Grassi et al. 2017, obr. 4–1)

Model pracuje se 4 hlavními aktéry, kterými jsou:

- Subscriber – Uživatel
- Credential Service Provider, CSP – Správa identit
- Relying party, RP – Poskytovatel služby
- Verifier – Verifikační služba[14]

Na obrázku jsou ještě znázorněni aktéři Applicant a Claimant, zde je ovšem rovnou považujme za jednoho aktéra – uživatele. Zároveň je důležité zmínit, že model rozděluje CSP, RP a verifikační službu na 3 odlišné aktéry, ale v praxi je možné že některé či všechny z nich budou jedním systémem. Proto je mezi CSP a verifikační službou přerušovaná čára, která znázorňuje logickou souvislost těchto dvou aktérů.

Pro lepší interpretování modelu je možné si jej rozdělit na dvě části. Levou, registrační, a pravou, autentizační.

[14] České překlady převzaty z (Pavlíček 2020)

Levá část se znázorňuje registraci, vydávání verifikátoru a ověřování identity. Obvyklá sekvence aktivit v této oblasti je:

1. Uživatel žádá CSP o registraci
2. CSP ověří identitu uživatele
3. CSP vydá uživateli verifikátor
4. CSP si povede záznam s informacemi o uživateli a jeho verifikátorech alespoň po dobu životnosti verifikátoru. Uživatel si uchová svůj verifikátor

Díky tomuto jednoduchému procesu nabude uživatel verifikátoru, který je možné použít pro budoucí autentizaci. Jako praktický příklad je možné uvést jednoduchou registraci, kde uživatel se chce registrovat do aplikace. Uživatel přijde na webové stránky aplikace a bude chtít otevřít registrační formulář. Aplikace mu jej zobrazí a zeptá se jej na email. Uživatel email vyplní a formulář odešle. Aplikace nyní musí ověřit, zdali tento email opravdu patří uživateli, a tak na danou adresu odešle email s odkazem, které musí uživatel otevřít, aby tím potvrdil aplikaci, že on opravdu je vlastníkem této adresy. Po tomto ověření již aplikace nechá uživatele např. nastavit svůj profil ale hlavně mu vydá nějaký verifikátor, který použije příště pro autentizování.[15]

V pravé části modelu je znázorněn primárně proces autentizace a autorizace, kde se uživatel ověřuje nově nabytým verifikátorem RP. Standardní kroky jsou následující:

1. Uživatel se chce autentizovat RP. Ta jej odkáže na verifikační službu
2. Uživatel poskytne verifikační službě svůj verifikátor
3. Verifikační služba komunikuje s CSP a ověří uživatelův verifikátor, případně si od CSP převezme informace o uživateli
4. CSP nebo verifikační služba potvrdí uživatelovu identitu RP a předá ji informace z minulého kroku
5. Je vytvořena autentizovaná relace mezi uživatelem a RP

Z pohledu dřívějšího příklad by zde šlo o uživatelovo přihlášení do aplikace. Uživatel by odeslal email a svůj verifikátor, heslo, aplikaci přes přihlašovací rozhraní. Aplikace by tyto údaje ověřila a uživatele autentizovala.

3.3 Typy autentizace

Účel této kapitoly je představit a obecně popsat principy jednotlivých typů autentizačních metod, z čehož poté vychází kapitola 5, která se věnuje těmto metodám přímo v prostředí Mendixu. Jsou zde rozebrány všechny metody definované v kapitole 2.1.

[15] V tomto příkladu by aplikace vystupovala jako jak RP, tak i CSP a verifikační služba.

3.3.1 Jméno a heslo

Autentizace pomocí jména a hesla patří mezi nejčastější způsoby autentizace. Verifikátor, tj. v tomto případě heslo, je typu „Něco, co znám“ a jedná se nejčastěji o uživatelem vytvořený libovolný řetězec znaků. Mnoho aplikací i uživatelů preferuje tento způsob, jelikož je velmi jednoduchý. Při registraci si uživatel vybere své přihlašovací jméno a heslo, které následně používá k následným přihlášením.

Problémem tohoto způsobu autentizace je jeho bezpečnost. Již dle (NIST 2020) spadá autentizace pomocí tzv. „Memorized Secrets“ mezi nejslabší metody autentizace. Hlavními problematickými oblastmi jsou zde uživatelé, kvalita hesel a ukládání hesel.

Uživatelé jsou zde problematičtí primárně tím, jak hesla tvoří a jak s nimi nakládají. V rámci registračního procesu je částečně možné ošetřit, jaké podmínky musí heslo splňovat, aby byla zajištěna určitá minimální kvalita hesla, avšak není možné ošetřit mnoho faktorů, jako je například používání stejného hesla napříč více aplikacemi[16]. NIST, OWASP a mnoho dalších institucí vydalo dohromady nespočet publikací, snažící se zvýšit kvalitu hesel. Obecně se shodují v následujícím:

- Heslo by mělo obsahovat alespoň 8 znaků
- Maximum znaků by nemělo být stanoveno nízko
- K heslu/účtu by neměly existovat záchranné/obnovovací otázky či nápovědy
- Heslo by mělo obsahovat malé i velké znaky, symboly či čísla
- Nepoužívat slova – hlavně ne slova spojená s aktuální osobou či aplikací

Jak již bylo ovšem nastíněno dříve, i velmi bezpečné heslo může být lehce kompromitováno pomocí phishingu, shoulder surfingu[17] či jiným špatným zacházením s heslem, jako je jeho znovu používání, sdílení, zapisováním na papírky/do mobilu/na plochu počítač apod.

Dalším problémem je, jak s heslem pracuje aplikace. Nejméně vhodnou metodou, je ukládat hesla v takové podobě, jak byly přijaty od uživatele, tzv. plaintext. Takovéto zacházení je vysoce rizikové, jelikož pokud se kdokoliv dostane do databáze či k souboru s hesly uživatelů, hned získá všechny přihlašovací údaje.

Lepším způsobem, jak hesla ukládat, je používat bezpečné, neprolomené hashovací algoritmy a následně přenášet a ukládat jen hashované hodnoty. Avšak ani tento přístup není dnes dostatečný, jelikož útočník může použít nástroje jako Rainbow Tables[18], pomocí kterých může stále nějaká hesla získat.

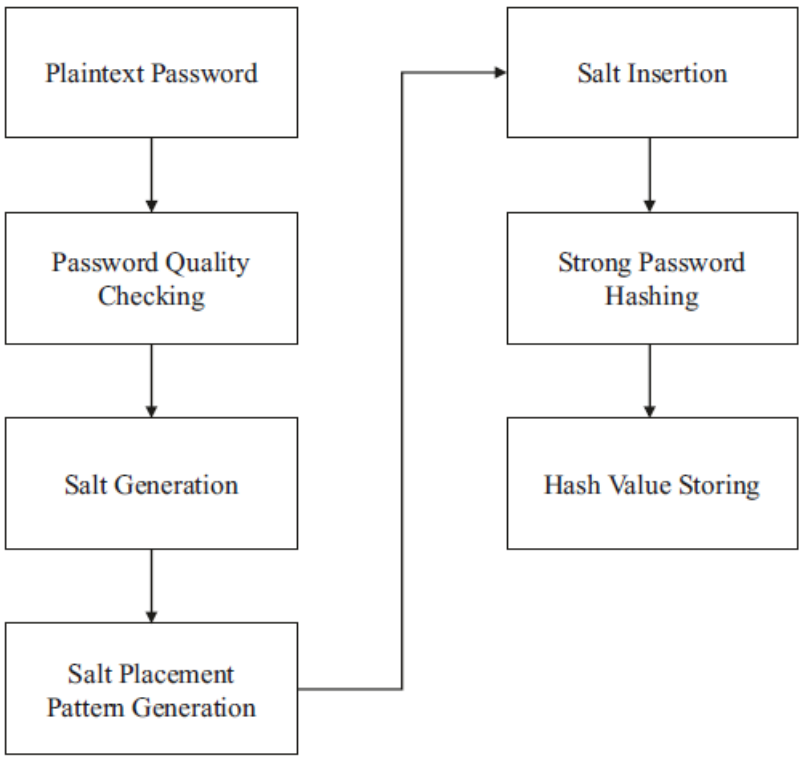
Aby byla aplikace schopná i tomuto odolat, je doporučené hesla tzv. solit, případně i pepřit. Solení spočívá v přidání náhodně vygenerovaného řetězce ke vstupnímu řetězci hashovací

[16] Dle (Vojinovic 2022) používá přes 51 % uživatelů stejné heslo pro firemní a osobní účty a přes 69 % uživatelů někdy sdílelo heslo s kolegou v práci.
[17] Technika sociálního inženýrství, kdy je heslo odpozorováno od uživatele tzv. „koukáním přes rameno“.
[18] Jedná se o tabulky předem vypočítaných hodnot, sloužící k rychlejšímu prolamování hashovaných hodnot.

funkce, který následně zajistí, že výsledný hash bude vždy jiný i v případě používání stejných hesel. V praxi tedy uživatel zadá heslo, které si přeje používat, systém k němu přidá sůl, výsledek použije jako vstup pro hashovací funkci a její výstup uloží do databáze společně i se solí. Dle nutnosti a požadované úrovně bezpečnosti je možné nastavovat množství soli či její umístění. Není totiž nutné, aby sůl byla umístěna jako prefix nebo sufix samotného hesla, ale je možné pracovat se vzory umístění soli, které na binární úrovni za pomocí XOR operace dokáží aplikovat sůl v daných místech a jinde ji vynechat. Díky tomu je možné docílit i výsledku, že stejné heslo se stejnou solí bude mít stále jiný hash (Boonkrong 2020).

Pepření je možné použít se solí jako další úroveň zabezpečení. Existuje více podob pepření, může jít o přidání další hodnoty, tj. pepře, k heslu před hashováním (Ibeakanma 2022), nebo o zašifrování výsledného hashe symetrickou šifrou, kde pepř bude sloužit jako klíč (OWASP 2022b). Hlavním rozdílem pepře od soli je fakt, že pepř je sdílen mezi vícero hesly najednou a zároveň není uložen ve stejné databázi jako je sůl a výsledné hashe. Cílem pepření tedy je zamezit útočníkům dostat se k původním hodnotám hesel v případě, že se dostanou k jejich hashím a soli.

Mnoho moderních hashovacích algoritmů, které se používají pro práci s hesly jako je například BCrypt již defaultně hashované hodnoty solí (Arias 2021). V dnešní době je důležité s hesly nakládat bezpečně, proto by měly být použity co nejmodernější hashovací algoritmy a určitě by se mněla hesla alespoň solit. Na Obrázek 2 je vyobrazen vzorový průběh uložení hesla, při kterém dochází jak ke kontrole kvality hesla z uživatelovy strany, tak i k solení, hashovaní a následnému bezpečnému uložení hash hodnoty hesla.



Obrázek 2: Proces uložení hesla (Boonkrong 2020, obr. 4–1)

3.3.2 OTP

Hesla na jedno použití, ang. One-Time Passwords (OTP) jsou počítači generovaná hesla, která uživatelé používají k jednorázové autentizaci. V praxi se často používají časově omezené varianty, tzv. Timed One-Time Passwords (TOTP), které mají pro vyšší bezpečnost navíc určitou dobu platnosti, po kterou je možné se autentizovat. OTP spadá do kategorie něco, co mám, jelikož jeho podstatou je přístup k zařízení generující či přijímající OTP k přihlášení.

OTP funguje principiálně stejně, jako hesla zmíněná v minulé podkapitole, jejich výhodou a důvodem užívání je vyšší bezpečnost, která se díky neustále měnícím heslům a jednorázovému použití drasticky zvyšuje (LeBlanc a Messerschmidt 2016). Zároveň je vyřazena většina hrozeb ze strany uživatele, jelikož ten heslo netvoří, neukládá a nemůže jej používat napříč více aplikacemi. Záleží ale ovšem na způsobu implementace OTP.

Existuje vícero typů a způsobů kategorizace OTP. Pro účely práce jsou přínosné primárně (Mathur 2022) a (OneLogin 2023), dle kterých je možné OTP rozdělit na generované na serveru a generované v aplikaci. OTP generované na serveru fungují jednoduše tak, že server posílá uživateli OTP v reakci na jeho akci. Uživatel tento kód obdrží pomocí emailu či SMS a zadá jej zpět do aplikace. OTP v tomto případě může být jakýkoliv náhodný řetězec.

Při generování OTP v aplikaci je princip komplikovanější. Uživatel musí mít prvně nainstalovanou mobilní či desktopovou aplikaci. Jako příklad je možné zmínit Google či Microsoft Authenticator. Následně je nutné propojit aplikaci se službou, u které se chce autentizovat. Toto propojení proběhne pomocí domluvení si tzv. secret key, statické hodnoty, kterou budou moci následně obě strany používat pro generování OTP[19]. Uživatel do tohoto nastavení obvykle nijak nezasahuje, jelikož často jsou tyto hodnoty nastaveny pomocí QR kódu, který jen naskenuje. Jakmile jsou aplikace propojeny, je možné začít OTP používat. Jelikož obě aplikace mají secret key, jsou schopné vygenerovat stejný OTP za stejných podmínek. Proto není nutné, aby server posílal OTP klientovi, ale pouze stačí, aby klient poslal svůj OTP, který následně server validuje pomocí jeho porovnání s OTP vygenerovaného na straně serveru.

V případě generovaní OTP v aplikaci jsou nejčastěji využívány 2 typy: HOTP (RFC 4226) a TOTP (RFC 6238). HOTP je OTP postavené na HMAC[20], kde ke generování OTP je použit ještě tzv. moving factor, což v praxi je číslo, které se zvyšuje s každým použitým a validovaným OTP. Jde tedy o dodatečnou úroveň ochrany, kde útočníkovi by pro generování stejného OTP nestačila pouze stejná konfigurace a secret key, ale musel by znát i aktuální číslo pořadí. TOTP funguje na podobném principu, avšak místo čísla používá aktuální čas. Standardem je používání 30 sekundových oken, tj. ne moc času pro útočníka, avšak dostatek času pro možné prodlevy ve komunikaci a na zpracování.

[19] Zároveň si mohou domluvit i další parametry, jako např. použitou hashovací funkci.
[20] Hash-based Message Authentication Code

Častokrát je možné se setkat s OTP jako s druhým faktorem autentizace, viz 3.3.7, avšak je možné, aby fungoval samostatně.

Nevýhodami OTP, se kterými se klasická hesla nepotýkala spočívají primárně v závislosti na zařízení generující či přijímající OTP. Pokud uživatel toto zařízení, např. mobilní telefon nemá, nebo je např. vybité, nemůže poté tento způsob autentizace využít. Zároveň jsou často mobilní zařízení náchylnější ke ztrátě a krádeži.

Další riziko může nastat, pokud se aplikace pro generování OTP nachází na stejném zařízení, které je používáno k autentizaci (OWASP 2021a).

Naopak kromě vyšší bezpečnosti oproti klasickým heslům může být ještě lepší uživatelská přívětivost a UX obecně, kdy je často možné setkat se s OTP v podobě QR kódů či sestrojené tak, aby je zařízení mohly automaticky načítat a importovat (LeBlanc a Messerschmidt 2016).

3.3.3 SSO

Tématice single sign-on, SSO, se věnuje tato a následující podkapitola, přičemž zde se nachází obecnější, zastřešující informace a v kapitole příští jsou rozebrány detailněji přístupy OIDC a OAuth. SSO je velmi široký pojem, označující širokou škálu přístupů a možností implementace. Zde bude ovšem kladen důraz na přístupy využitelné v Low-Code platformě Mendix.

Podstatu SSO vhodně popisuje následující úryvek z dokumentace Azure Active Directory od firmy Microsoft:

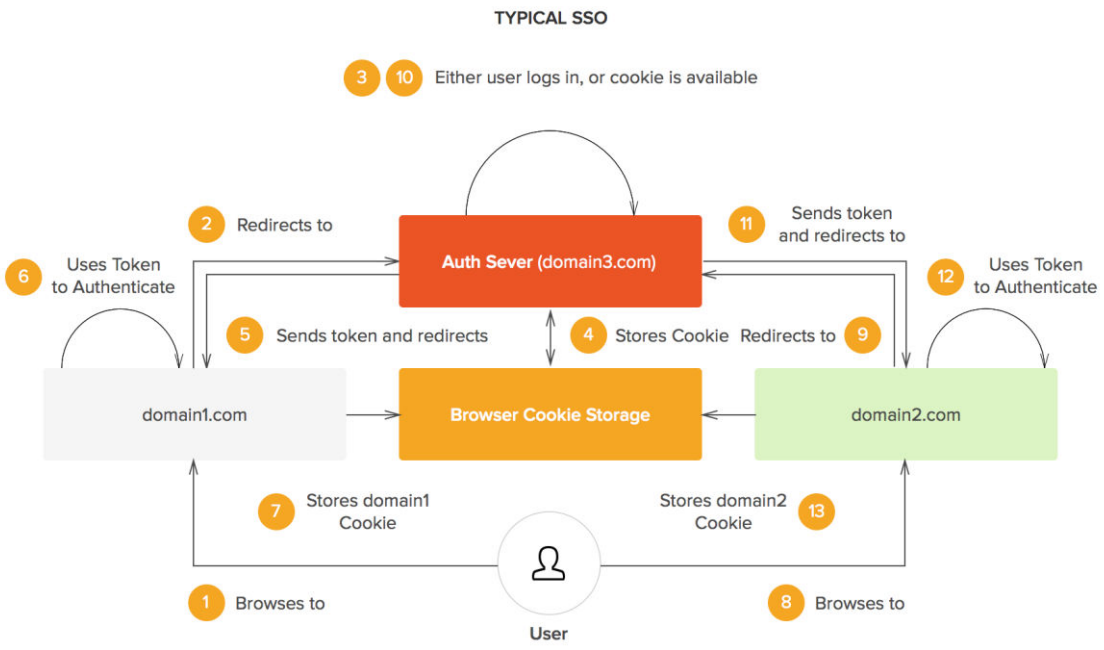
Single sign-on is an authentication method that allows users to sign in using one set of credentials to multiple independent software systems. Using SSO means a user doesn't have to sign in to every application they use. With SSO, users can access all needed applications without being required to authenticate using different credentials. (Microsoft 2022f, odst. 1)

Přesně jak autoři zmiňují, SSO je autentizační metoda, která umožňuje uživateli přihlásit se v jedná aplikaci a následně použít tuto existující relaci pro autentizaci u jiných aplikací. Ve výsledku se tedy uživatel přihlásí jen jednou, jak vyplývá z názvu metody, a tím se dokáže autentizovat u více aplikací. Tato praktika je dnes standardní u středních a velkých firem, jako je například již dříve zmíněný Microsoft, nebo Google, Amazon. Zároveň jej využívá mnoho bank či interních firemních systémů.

Samotná prvotní autentizace probíhá například pomocí hesla, či jiné autentizační metody, v tomto se od ostatních metod příliš neliší. Její hlavním rozdílem je schopnost komunikovat autentizované připojení a autentizovat uživatele u jiných aplikací. V základu tedy jde o přihlášení pomocí něco, co mám verifikátoru, jako může být například heslo, avšak primárním přínosem této metody je až dříve zmíněná autentizace u ostatních aplikací.

Níže na Obrázek 3 je obecně znázorněn jedna z mnoha možných implementací SSO. V tomto případě se uživatel postupně snaží přihlásit na dvě různé aplikace, běžící na adresách

domain1.com a domain2.com, které využívají stejný autentizační server (na adrese domain3.com). V tomto případě autor schématu pracoval s tzv. JWT[21] tokeny. Tento token by si prohlížeč uživatele uložil po úspěšné autentizaci a následně jej využil pro další autentizování, za předpokladu nepřesáhnutí předem dané trvanlivosti. Pokud by tedy šlo o jednotnou relaci, uživatel by navštívil domain1.com, odsud by byl přesměrován na autentizační server a po autentizaci si uložil JWT. Následně by navštívil domain2.com použil zde JWT pro autentizování, bez vyplňování jakýchkoliv údajů jako u minulé aplikace (Peyrott 2022).



Obrázek 3: Příklad obecného SSO (Peyrott 2022)

Hlavní výhodou SSO je pro uživatele komfort, který vychází z jednoduchosti autentizace a možnosti používat jeden účet k více aplikacím najednou. Pro poskytovatele je zde znatelnou výhodou centralizace uživatelských účtů a dat[22], zvýšení bezpečnosti[23] a v neposlední řadě i usnadnění práce vývojářů nových systémů (Kumar 2022).

Typy

Na začátku minulé kapitoly bylo zmíněno, že SSO označuje širokou škálu přístupů a možností implementace. V této kapitole budou některé z nich krátce představeny.

[21] JSON Web Token
[22] A časové či finanční výhody plynoucí z efektivnější správy a údržby.
[23] Např. díky centralizovanému Access managementu či možnosti použití detailních auditních či bezpečnostních opatření.

Vybrané příklady a jejich popis byly převzaty od poskytovatelů SSO, jako je Microsoft, Okta nebo OneLogin.

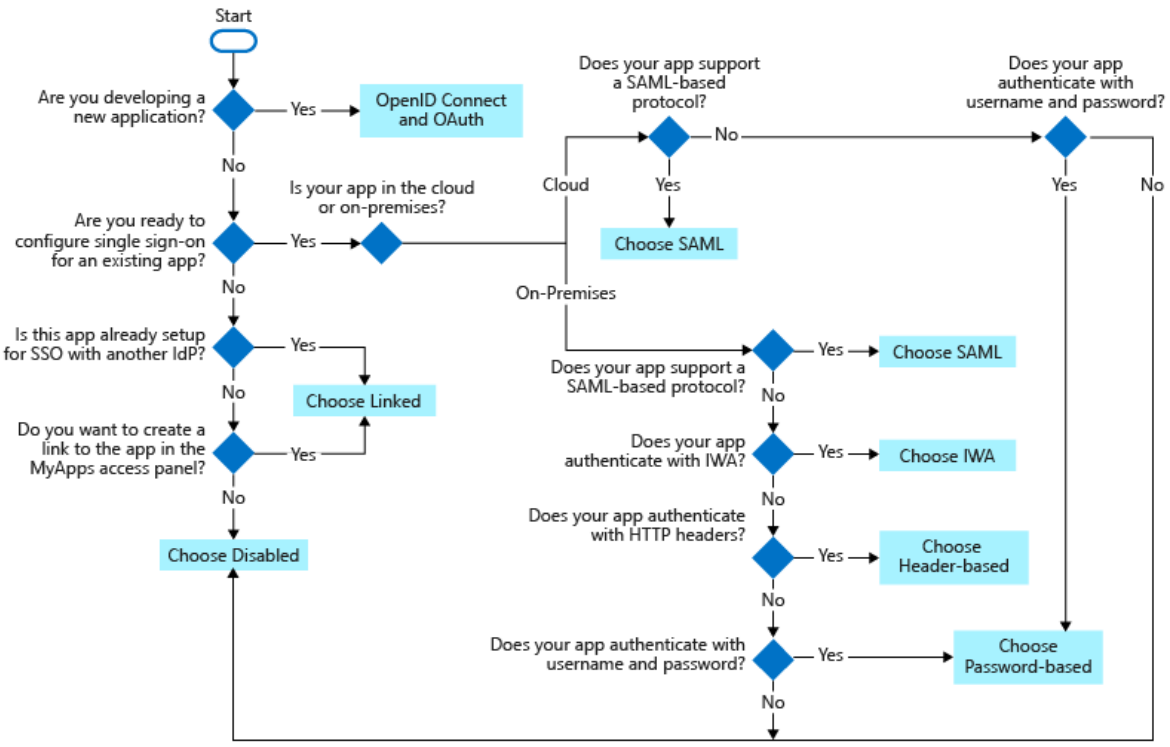
Prvním typem, se kterým je možné se nejčastěji setkat, je SSO implementované za pomoci OpenID Connect a protokolu OAuth 2.0. Tomuto typu je věnována celá podkapitola 3.3.4.

Dalším typem je SAML SSO. Je založen na XML standardu SAML, Security Assertion Markup Language, který se používá pro výměnu dat mezi identity providerem a service providerem. Rozdíl oproti prvnímu typu je jiná struktura komunikace mezi zúčastněnými stranami. Nutné zmínit, že tento způsob SSO není doporučován mnoha poskytovateli, jako např. Microsoftem či společností Okta – obě doporučují používat pokud možno OpenID SSO (Guevara 2021) (Microsoft 2022b).

Je možné také zmínit Password-Based SSO, které funguje téměř na principu správce hesel. Tento typ SSO funguje způsobem, že si uloží přihlašovací údaje uživatele do vlastní databáze při prvním přihlášení, a následně jej automaticky přihlašuje při dalších návštěvách (Microsoft 2022a).

(OneLogin 2022b), dále ještě zmiňuje existenci SSO používajících protokoly LDAP či Kerberos. Microsoft jich rozlišuje další 3, které nebyly zmíněny.

Na závěr této kapitoly je zde umístěn flowchart, který firma Microsoft používá k rychlé ilustraci průběhu výběru SSO, viz Obrázek 4. Tento obrázek vhodně ilustruje, že existují různé rozdíly a možnosti, které je nutné brát v potaz při porovnávání či používání SSO. Zároveň je v něm možné najít celkem 6 typů SSO.



Obrázek 4: Typy SSO (Microsoft 2022c)

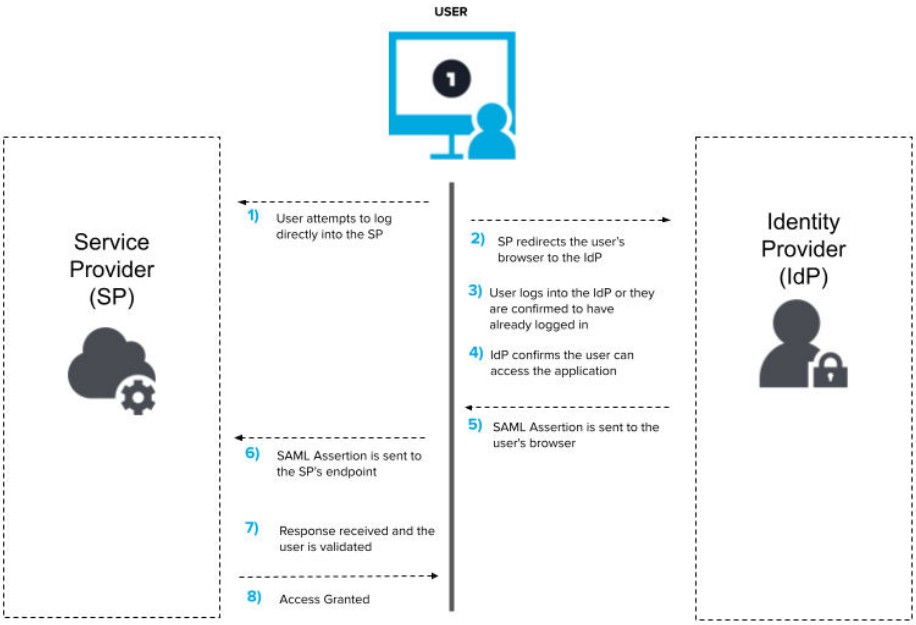
3.3.4 OIDC a OAuth

OIDC, OpenID Connect, patří mezi nejmodernější a nejrozšířenější typ SSO, a právě proto je mu věnována celá tato kapitola. Avšak aby mohl být celkově představen, bude nutné detailněji rozebrat i protokol OAuth 2.0 a standard SAML, které již byly částečně zmíněny dříve.

SAML

Security Assertion Markup Language, SAML 2.0, je standard pocházející z roku 2005. Ve své době poskytl řešení pro ranné SSO řešení, které potřebovaly vyřešit problém propojení externích SAAS aplikací s centrálním autentizačním řešením (Wilson a Hingnikar 2019).

SAML definuje způsob komunikace mezi tzv. Service Providerem, SP, a Identity Providerem, IdP, a to jak v podobě celé komunikace, tak i zpráv. SP je obecně systém, který uživatel chce používat a který je propojen s IdP pro účely autentizace. IdP je poté systém, který je používán k autentizování uživatelů a jejímu potvrzení u SP. Ke komunikaci jsou použity klasické http či SOAP requesty a obsahy zpráv jsou posílány ve formátu XML. Obecné schéma přihlášení za pomoci SAML je zobrazeno níže na Obrázek 5.



Obrázek 5: Schéma SAML (Townsend 2021)

SAML Assertion, zmíněná v krocích 5) a 6) je obecný název pro zprávy, předávané od IdP ke SP, které obsahují informace o autentizaci či autorizaci, nebo obecné atributy. Může tedy například jít o informace o tom, jestli je uživatel přihlášen nebo ne.

Hlavní výhodou SAML je možnost centralizace uživatelských dat a access managementu obecně. Jde tedy o funkční a spolehlivé SSO řešení, které je ještě dnes jedno z nejpoužívanějších. Na druhou stranu je SAML spíše firemně-orientován a zároveň je složitější jej implementovat oproti modernějším řešením jako je OIDC, jelikož je více

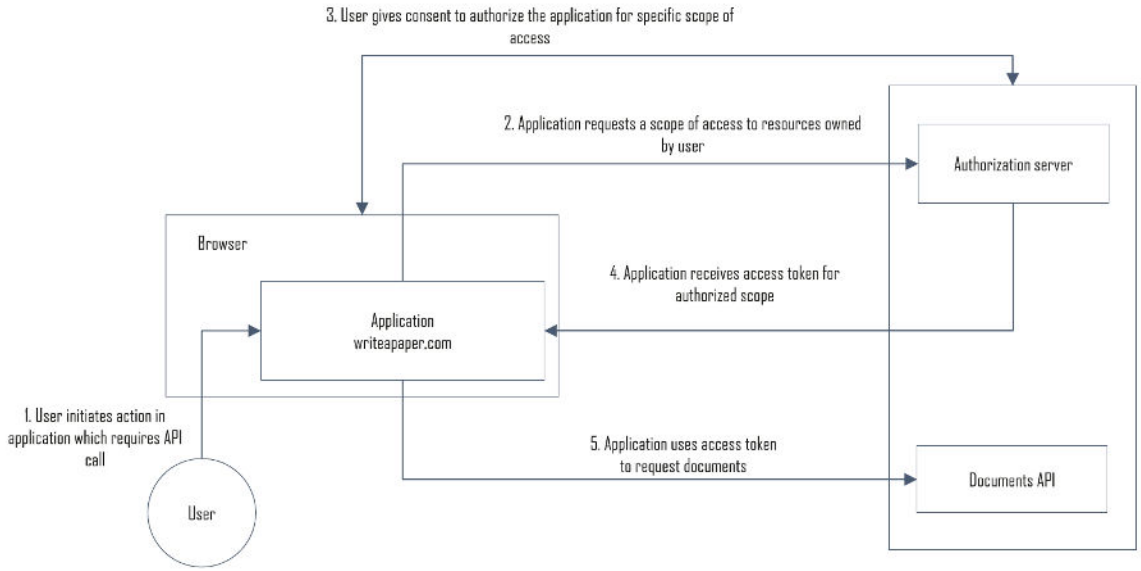
náchylný k chybám. Z bezpečnostního hlediska SAML také nestačí na OAuth 2.0 a OIDC, protože je obtížnější jej implementovat bezpečně a zároveň se musí potýkat s hrozbami, které OAuth či OIDC řešit nemusí, jako jsou zranitelnost XML či jeho obtížnější šifrování oproti JSON (OneLogin 2022c).

OAuth 2.0

Předtím, než bude věnován prostor OIDC, je vhodné zabývat se protokolem OAuth 2.0, na kterém následně OIDC stojí a který rozšiřuje. Kapitola se zabývá výhradně protokolem ve verzi 2.0, jedná se o verzi nejpoužívanější, avšak existují již první kroky vedoucí k verzi 2.1 (OAuth 2022).

OAuth 2.0 is a delegation protocol, a means of letting someone who controls a resource allow a software application to access that resource on their behalf without impersonating them (Richer a Sanso 2017, s. 3)

OAuth je autorizační protokol[24] pocházející z roku 2012. Jeho primárním use case je umožnění uživateli, v OAuth terminologii resource owner, povolit aplikaci, aby nějak pracovala s jeho daty na odděleném serveru, tzv. resource server. Pro lepší znázornění je na Obrázek 6 znázorněno jedno z možných OAuth řešení.

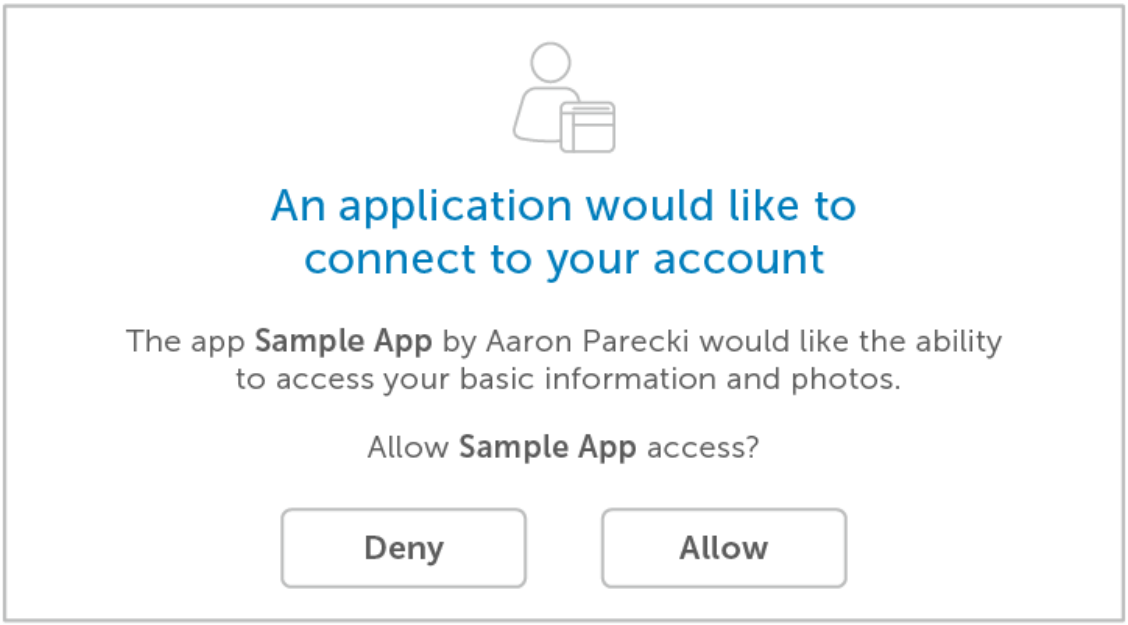


Obrázek 6: Příklad OAuth (Wilson a Hingnikar 2019, obr. 5–3)

Na obrázku je možné vidět příklad, kdy se uživatel snaží přes aplikaci writepaper.com získat dokumenty ze serveru, na kterém běží autorizační server a API s dokumenty. Poté, co uživatel o dokumenty zažádá, je vyzván k autentizaci, po které se ho server zeptá, zdali povoluje aplikaci writepaper.com, aby získala přístup k dokumentům. Pokud uživatel

[24] Někdy nazýván i delegační protokol (Richer a Sanso 2017) či autorizační framework (Wilson a Hingnikar 2019)

povolí, je mu zaslán access token, který následně může být použit k získání daných dokumentů. Příklad, jak může vypadat žádost o povolení je uveden na Obrázek 7.



Obrázek 7: Příklad OAuth žádosti o povolení (Okta 2022a)

Ve výsledku uživatel se uživatel autentizoval u serveru a následně povolil aplikaci, aby přistoupila k datům na serveru pomocí tokenu. Přitom během komunikace byly pevně stanoveny tzv. scopes, které definovaly, k jakým zdrojům získává aplikace možnost přistupovat.

Ačkoliv OAuth potřebuje autentizaci pro správný chod, sám není autentizačním protokolem. Je možné pomocí něj autentizační protokol vytvořit, viz následující kapitola, ale on sám nepřenáší žádné informace o uživateli a pouze pracuje s tokeny. Obecně jde jen o protokol, definující, jak token získat a jak jej použít. OAuth zároveň nepřikazuje, jak má komunikace či tokeny vypadat, čímž se odlišuje od SAML z minulé kapitoly (Richer a Sanso 2017).

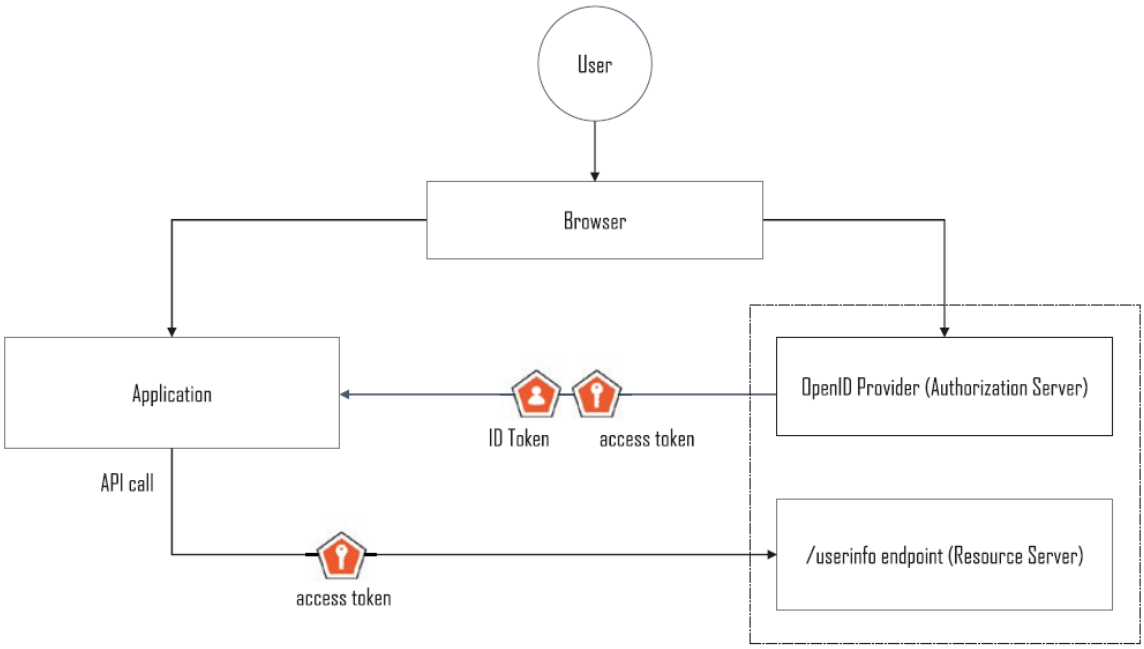
OIDC

OpenID Connect, OIDC, je autentizační framework, často prezentován jako autentizační vrstva stojící nad OAuth 2.0. OIDC vznikl v únoru 2014, kdy byl vytvořen standard OIDC společností OpenID Foundation (LeBlanc a Messerschmidt 2016).

Oproti OAuth 2.0 dokáže OIDC bezpečně předávat informace o identitě autentizovaného uživatele ve standardizovaném formátu, díky čemuž se dokáže fungovat jako SSO pro mnoho situací. Zároveň není tolik interně orientován jako SAML, a v praxi je možné implementovat SSO za pomoci sociálních sítí či služeb jako je Google, PayPal, Yahoo a mnoho dalších, díky čemuž je možné budovat aplikace s SSO využitelným pro širokou veřejnost (Wilson a Hingnikar 2019).

OIDC provides the web single sign-on benefits that were attractive in SAML 2.0 and, when combined with OAuth 2.0, provides a solution with authentication as well as the API authorization capabilities needed by modern applications (Wilson a Hingnikar 2019, s. 26).

Průběh autentizace OIDC je podobný, jako v průběh OAuth 2.0. Schéma je znázorněno na Obrázek 8. Ve chvíli, co uživatel navštíví aplikaci, bude přesměrován na autentizační server – ve schématu nazván OpenID Provider. Odkud, po autentizaci, bude přesměrován zpět do aplikace, spolu s autorizačním kódem, který může aplikace využít pro získání ID tokenu a access tokenu z enpointů poskytovatele OpenID. Šipka uprostřed schématu tuto komunikaci zjednodušuje a místo několika requestů zobrazuje pouze jeden.



Obrázek 8: OIDC Autentizace (Wilson a Hingnikar 2019, obr. 6–1)

Rozdílem od OAuth 2.0 primárně je, že zde přibyla autentizační vrstva, která pracuje s tzv. ID Tokenem, který přenáší informace o tom, zdali byl uživatel autentizován, a zároveň navíc pracuje s tzv. UserInfo endpointem, který může aplikace využívat, aby získala informace o uživateli. Stejné ovšem mají, jak je pracováno s access tokenem a autorizaci obecně.

Od SAML se OIDC liší tím, kdo autentizuje uživatele a jak je s touto informací následně zacházeno. V případě SAML, aplikace (SP) přesměruje uživatele na předem definovaný server pro autentizaci (IdP), které následně vrátí assertion zpět aplikaci. Je zde tedy důležitý vztah a důvěra mezi SP a IdP. V případě OIDC je autentizace prováděna veřejnou aplikací třetí strany. Aplikace může například povolovat pro přihlášení Facebookový účet, Google účet, nebo Microsoft účet – uživatel si vybere jednu z možných variant a následně je přesměrován na její login stránku. Po autentizaci se odešle aplikaci zpět autorizační kód, který může aplikace využít pro získání tokenů a tím získá všechny informace co potřebuje.

Zároveň je OIDC jednoduší jak SAML a k přenosu dat používá méně komplexní Json a JWT, které jsou jednodušší na zpracování a díky menší velikosti méně namáhají aplikace. Zároveň

je jednodušší implementovat OIDC bezpečně, jelikož pro šifrování je možné použít HTTPS či SSL (Brown 2022).

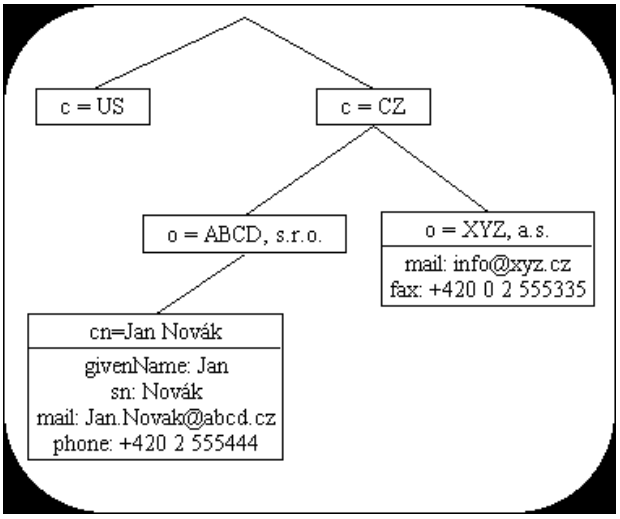
Velkou výhodou OIDC oproti SAML a jiným SSO řešením, je již dříve zmíněná uživatelská přívětivost, vycházející z možnosti vyžití jiných osobních účtů. Např. účtů sociálních sítí, či populárních služeb či aplikací, emailových klientů apod. Díky tomu je dnes OIDC jedním z nejvyužívanějších a nejpopulárnějších SSO řešení.

3.3.5 LDAP

LDAP is the Lightweight Directory Access Protocol. It's a standards-based protocol that sits on top of TCP/IP and allows clients to perform a variety of operations in a directory server, including storing and retrieving data, searching for data matching a given set of criteria, authenticating clients, and more (LDAP 2018, sek. Learn about LDAP).

Protokol LDAP je standardizovaný protokol, používaný pro komunikaci a interakci s adresářovými servery. Jedním z jeho použití je i autentizace uživatelů, díky čemuž je možné efektivně budovat SSO založené na LDAP.

Vznikl v první polovině devadesátých let, kdy byl vytvořen jako odlehčená verze mezinárodních standardů X.500[25]. Adresáře, se kterými protokol pracuje, jsou organizovány ve stromové struktuře a obsahují záznamy. Nejvyšší záznam, se nazývá kořen a obvykle jde o reprezentaci společnosti, která vlastní daný adresář. Níže položené záznamy poté reprezentují oddělení či jiné uskupení, a poslední, tzv. listové, záznamy již reprezentují samotné objekty, osoby či zdroje. Každý záznam může obsahovat atributy, do kterých je možné ukládat samotné informace, jako jméno, email apod. Schéma jednoduchého LDAP stromu je znázorněno na Obrázek 9 (Oracle 2022a).



Obrázek 9: LDAP strom (Zapletal 2000, obr. 1)

[25] Sada standardů pro globální komunikaci s adresářovými servery a službami (Oracle 2022c).

Takto uložené záznamy je poté možné dotazovat, tvořit či měnit pomocí standardizovaných příkazů. Ty jsou prováděny ve standardním modelu client-server a request-response a je možné je provést přímo přes TCP nebo šifrovat pomocí SSL/TLS (Lavender a Wahl 2003).

LDAP autentizace spočívá obvykle v ověření uživatelského verifikátoru, např. hesla, a přihlašovacího jména proti uživatelským údajům uloženým v databázi. Přihlášení se provádí se pomocí specializovaného příkazu „bind" a je možné použít dva různé způsoby autentizace. Prvním je jednoduchá autentizace pomocí hesla[26], kdy se na server pošle heslo a přihlašovací jméno jako obyčejný textový řetězec. Druhým je SASL[27], který umožňuje specifikovat tzv. SASL Security mechanism, který bude použit pro autentizaci. SASL umožňuje používat autentizační metody, které nepracují s hesly a obecně je považován za bezpečnější.

Oracle LDAP SASL dokumentace jmenuje tyto SASL bezpečnostní mechanismy:

- *Anonymous (RFC 2245)*
- *CRAM-MD5 (RFC 2195)*
- *Digest-MD5 (RFC 2831)*
- *External (RFC 2222)*
- *Kerberos V4 (RFC 2222)*
- *Kerberos V5 (RFC 2222)*
- *SecurID (RFC 2808)*
- *Secure Remote Password (Using the Secure Remote Password (SRP) Protocol for TLS Authentication)*
- *S/Key (RFC 2222)*
- *X.509 (draft-ietf-ldapext-x509-sasl-03)*

(Oracle 2022b, kap. SASL Authentication)

3.3.6 Biometrické přihlášení

Biometrie je v této práci první a jedinou metodou autentizace z kategorie verifikátorů něco, co jsem. Zároveň se jedná o nejmladší ze zmiňovaných metod, kdy se její popularita a používanost rapidně zvýšila během poslední dekády, a to primárně díky nárůstu užívání mobilních telefonů a smartphonů. Jako další důvod je ovšem možné zmínit, že biometrické přihlašování je bezpečnější, jak přihlašování za pomocí verifikátorů kategorie něco, co vím (Boonkrong 2020).

Obecně je možné biometrickou autentizaci popsat jako autentizaci pomocí biologických, či behaviorálních charakteristik uživatele. Příkladem biometrických charakteristik může být snímek prstu, sítnice oka, cév, DNA apod. Jako příklad behaviorálních charakteristik je možné uvést způsob užívání dotykové obrazovky[28], způsob psaní na klávesnici či aktivita

[26] V (Lavender a Wahl 2003) nazýváno *simple password-based authentication*.
[27] Simple Authentication and Security Layer
[28] Např. jakou oblast uživatel využívá, jaké vznikají vzory pohybu ruky apod.

myši či ukazatele (OneLogin 2022a). S behaviorálními biometrickými charakteristikami se ovšem průměrný uživatel nejspíše nepotká, a primárně bude zvyklý na dva způsoby biometrické autentizace, a to na snímek prstu a obličeje.

Jak biometrická autentizace funguje je velmi různorodé. Záleží na typu a na implementaci. Je možné například ukládat celý obrázek, otisk, či jiná data v celku a následně je ověřovat při autentizaci. Častější přístup ovšem bývá ukládání jen několika klíčových bodů či oblastí, kdy pak při autentizaci dochází jen k párování těchto detailů a ověřování, zdali jde o stejnou osobu (Boonkrong 2020).

Výhody biometrie je primárně její unikátnost. Zároveň je zde eliminováno mnoho problémů, se kterými se potýkají hesla, jako je špatná tvorba a úschova hesel, nebo také nutnost pamatování si vícero komplexních hesel či jejich časté obměňování. Dokáží tedy být bezpečnější i snadnější na používání.

Biometrie avšak není neprolomitelná. Mobilní zařízení často pracují s částečnými skeny otisků, které je možné obelstít. Zařízení skenující obličej může být možné zase obejít pomocí masky[29]. Z uživatelské strany může zároveň dojít k problémům, kdy zařízení nemusí dokázat rozpoznat správného uživatele.

Další problémy, které s biometrickou autentizací mohou nastat, je například zabezpečení ukládání biometrických dat, které mohou být pod různou právní ochranou. To dokáže býti velmi problematické zvláště pro nadnárodní společnosti. Pokud by takováto data byla potenciálně ukradena, uživatel nemůže proti této situaci nijak bojovat, jelikož svůj otisk prstu si změnit nemůže. Následně jsou zde etické problémy, kdy zařízení nedokáží stejně efektivně rozpoznávat různé národnosti, pohlaví nebo rasy (OneLogin 2022a).

3.3.7 Více faktorová autentizace

Více faktorová autentizace spočívá v použití více faktorů (verifikátorů) pro úspěšné přihlášení. Typická implementace v prostředí webových aplikací je přihlášení pomocí hesla a následné vyplnění OTP, které přijde uživateli na mobil či do emailové schránky, viz.

[29] Tzv. „presentation attacks“ – útoky, které mají za cíl zasahovat do provozu biometrického systému (NIST 2020).

Obrázek 10: Příklad dvou faktorové autentizace (Boonkrong 2020, obr. 6–1)

Důvodem vzniku a používání více faktorové autentizace je potřeba vyššího zabezpečení. Dva faktory, např. dříve zmíněné heslo a OTP dokáží razantně zvýšit bezpečnost a objektivnost identifikace při autentizování. Potenciální útočník musí získat dva jiné údaje, aby mohl účet kompromitovat a v případě, že by se dostal k heslům uživatelů v databázi, nebude schopen se bez dalších faktorů přihlásit. Jde tedy o autentizační metodu, která využívá dvou či více metod pro přihlášení uživatele. Zároveň ji jde chápat jako přidanou úroveň bezpečnosti (Boonkrong 2020).

Je vhodné, aby jednotlivé verifikátory byly jiného charakteru, tj. aby nebyly použity dva verifikátory kategorie „něco, co mám“ či „něco, co vím“, ale aby byl použity jejich kombinace. Důvodem je vyšší výsledná úroveň bezpečnosti. Vhodným příkladem je třeba dříve zmíněná kombinace hesla a OTP, dalším by mohl být např sken otisku prstu a OTP, či hardware token a heslo (Wilson a Hingnikar 2019).

V praxi je více faktorová autentizace používána spíše jen za určitých podmínek či v určitých situacích. Může sloužit jako dodatečné zabezpečení kritických operací, pro zabezpečení přístupu k citlivým datům či jako doplněk administrátorské autentizace. Často je využívána v oblastech finančních služeb, zdravotnictví, vzdělávání či na sociálních médiích.

Jako praktický příklad je možné uvést dřívější způsob přihlašování do online bankovnictví UniCredit. Každý klient, který měl v bance zřízen účet obdržel zařízení v podobě malé kalkulačky a čtyřmístný PIN. Aby se mohl do bankovnictví přihlásit, musel do této kalkulačky zadat svůj PIN, aby mu byl vygenerován časově omezený OTP, který použil k autentizaci. Uživatel pro přihlášení musel tedy disponovat několika verifikátory: Zařízením, PIN, výsledný OTP aby se mohl přihlásit pod svým číslem do bankovnictví (UniCredit Bank 2019).

4 Mendix

4.1 Low-Code

Low-Code je specifický přístup k vývoji aplikací. Neexistuje pro něj jedna ucelená definice, proto pro vysvětlení jeho podstaty využiji definic hlavních představitelů na dnešním trhu, především Mendix (Mendix 2022i), OutSystems (OutSystems 2022) a MS Power Apps (Microsoft 2022d).

Název Low-Code se uchytil kolem roku 2014, kdy jej použila firma Forrester v reportu aktuálních trendů (Richardson a Rymer 2014), avšak počátky Low-Code je velmi obtížné definovat, což je způsobené i nepřesnými definicemi, často upravené pro své účely samotnými poskytovatelskými firmami. Osobně bych za dobu vzniku Low-Code platforem považoval počátky 21. století (kolem 2000–2005), kdy začaly vznikat platformy, které dominují trhu dnes.

Je to vývoj založený na extenzivním užívání modelů, na tzv. drag & drop prostředí, a na bohatých grafických a vizuálních prvcích. Tímto se výrazně liší od jiných jazyků, ve kterých je standardní psaní textu pro definici tříd, funkcí či vzhledu stránek. Notace použité pro grafické znázornění komponent jsou velmi podobné například EPC[30] či BPMN[31] notaci, nebo UML[32] diagramům či jiným, obecně známým a užívaným způsobům v modelování. Ukázky, jak vypadá zápis logiky/funkcí (viz Obrázek 11) a jak vypadá tvorba stránek (viz Obrázek 12), se nachází na konci kapitoly. Je ovšem samozřejmě možné psát i tradiční kód.

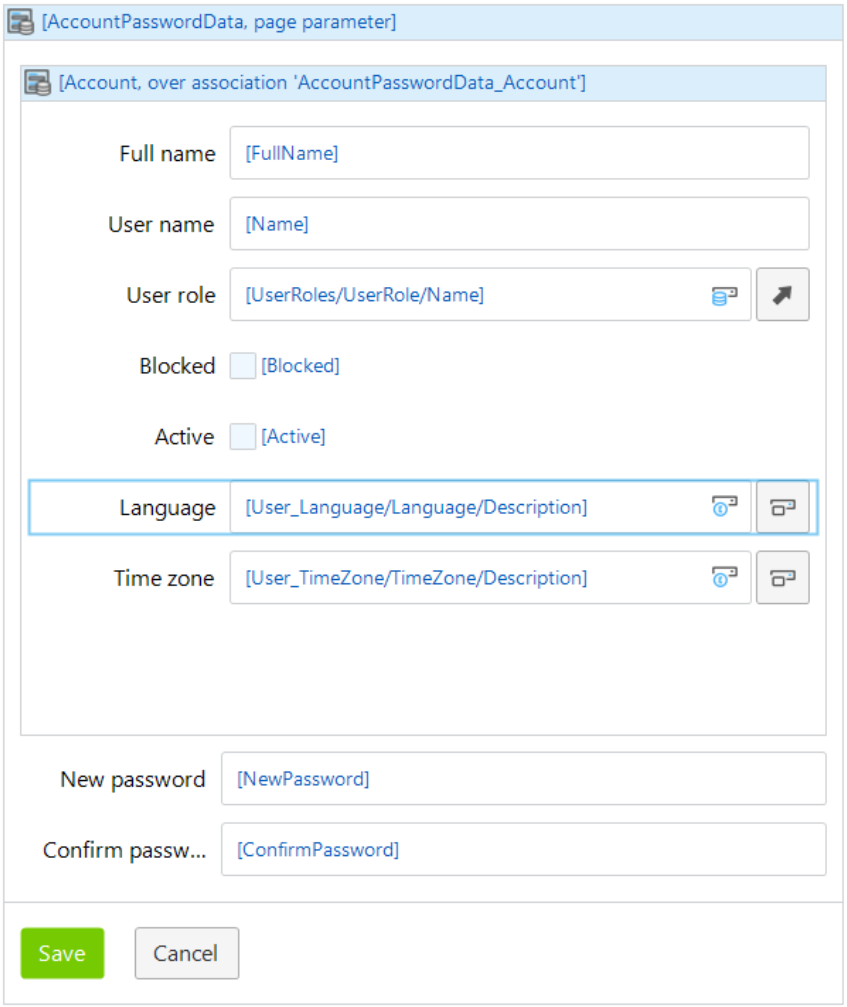
Zároveň většina Low-code platforem funguje na modelu PaaS, který umožňuje rychlé nasazení a aktualizace vyvíjených aplikací spolu se zabudovanými verzovacími a kolaboračními nástroji, které umožňují práci na několika vývojových větvích, snadné sledování změn a zálohování. Často jsou využívány agilní metodiky, jako je například Scrum a některé z platforem mají jejich principy zabudovány přímo do interních procesů. Cloudové řešení zároveň poskytují snadné škálovací řešení a integrace napříč aplikacemi.

Dalším často zmiňovanou vlastností Low-Code je jeho přehlednost, snadná dokumentovatelnost, lehkost používání a nenáročnost na technické či programovací znalosti. Často se zmiňují tzv. Citizen či Business vývojáři – jedná se o lidi z byznysu, ne o specializované vývojáře, kteří by měli být schopni navrhovat a programovat jednoduché aplikace. Některé firmy pro ně proto poskytují i zjednodušené platformy, aby pro ně vývoj co nejvíce usnadnily.

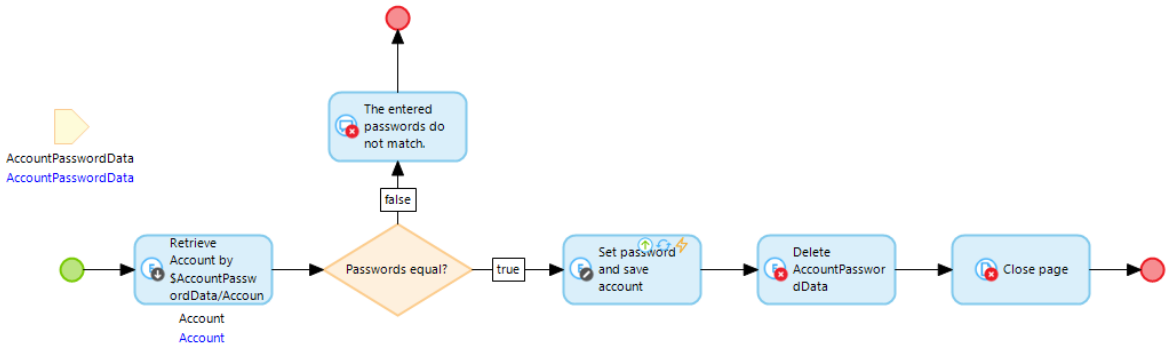
[30] Event Process Chain, viz (ARIS 2011).
[31] Business Process Model and Notation
[32] Unified Modelling Language (uml-diagrams.org 2022).

Jako poslední vlastnost Low-Code bych chtěl uvést vestavěnou správu životního cyklu aplikace. Je možné zde vyvinout, testovat, nasadit, monitorovat, aktualizovat, integrovat aplikace různých typů a pro různá zařízení.

Obecným cílem Low-Code, vycházejícím z jeho charakteristik, je zvýšit agilitu, snížit náklady, zvýšit produktivitu, zlepšit UX, zefektivnit risk management a Governance, usnadnit změny, a urychlit digitální transformaci (Appian 2021). Tyto cíle dokáží platformy naplňovat – jako příklad je možné uvést studii (Creatio 2021), zkoumající využívání Low-Code a No-Code platforem ve světových IT firmách mezi roky 2020 a 2021, která uvedla, že jen v 5% případů, nedošlo ke zrychlení vývoje za použití Low-Code, přičemž nejvíce firem zaznamenalo zrychlení vývoje o 41% až 60%. Jiná studie (Sauce Labs 2021) zase uvedla, že v 39,2% došlo za použití Low-Code ke zkvalitnění výsledného software, ve 26,3% byla kvalita obdobná jako u tradičního programování a ve 23,6% došlo ke zhoršení kvality (zbylých 10,9% nedokázalo posoudit).

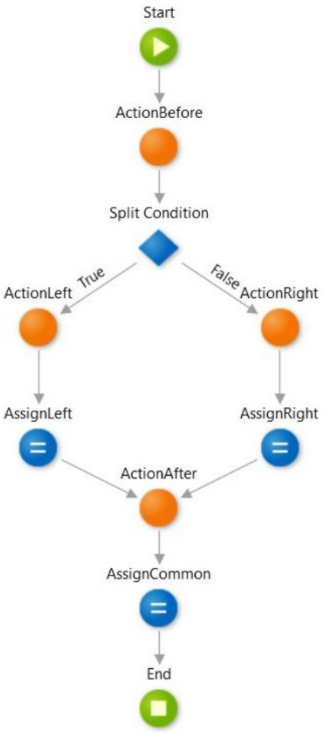


Obrázek 11: Ukázka tvorby stránky v Low-Code (zdroj: Autor)



Obrázek 12: Ukázka tvorby logiky v Low-Code (zdroj: Autor)

Poznámka k ukázkám: Jedná se o velmi jednoduchou stránku pro tvorbu nového účtu, kdy ve stránce jsou uživatelem vyplněny všechny potřebné údaje, a po stisku tlačítka „Save" je spuštěna funkce na druhém obrázku. Tato funkce prvně z databáze nový, či editovaný účet pomocí vazby z pomocné entity „AccountPasswordData" (která do logiky vstupuje jako parametr) a následně zkontroluje shodu hesel. Pokud hesla nesedí, ukáže se chybová hláška uživateli, pokud jsou v pořádku, účet se uloží do databáze, smaže se pomocná entita a zavře se editační stránka. Příklad je převzat ze standardního „Administration" Mendix modulu. Ostatní Low-Code platformy používají podobný zápis, např. níže je ukázka z platformy OutSystems, jiné zase ale jiný, jak tomu je v případě Power Apps.



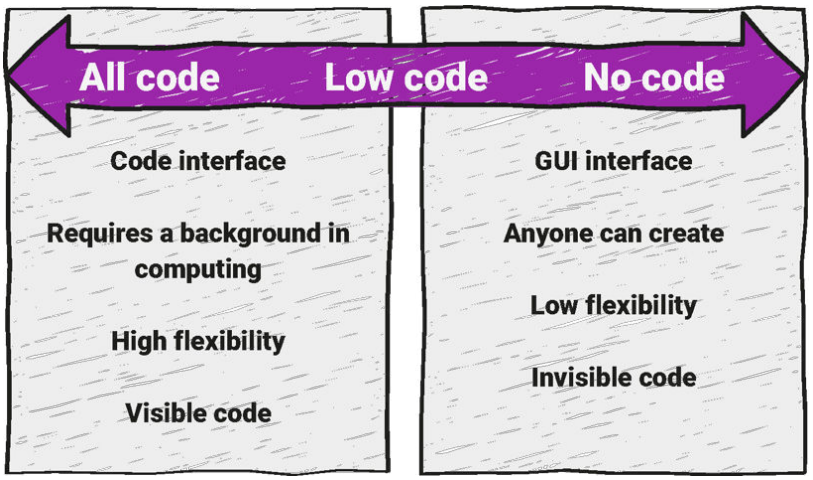
Obrázek 13: Ukázka tvorba logiky v Low-Code 2 (Mota 2020)

4.1.1 Low Code vs No Code

Rozdíly tradičního programování a Low-Code byly poznamenány již v minulé kapitole, kde bylo zmíněno, že mezi hlavní rozdíly patří programování pomocí grafických prvků, oproti

psaní kódu, drag & drop prvky, citizen developers atd. Tato definice ovšem není dostačující, jelikož existuje další oblast programování, nazývaná No-Code[33].

No-Code je velmi podobný Low-Code v mnoha ohledech, a jeho primárním rozdílem, již vycházejícím z názvu, je nepsání žádného kódu. Jako vhodná ilustrace poslouží osa na Obrázek 14, která zobrazuje Low-Code, jako mezistupeň, či prostředníka „All code“, tj. tradičního programování, a No-Code.



Obrázek 14: All Code vs No Code (LXA 2022)

Pro definování Low-Code je tedy nutné si uvědomit, že zde ještě určité možnosti ke psaní tradičního kódu jsou a že existují variace programování, které tento koncept zjednodušení vývoje berou mnohem více do extrému.

Nakonec této kapitoly bych ještě chtěl vyzdvihnout, že významným rozdílem mezi Low-Code a No-Code je i celkový přístup k vývoji a použití. V případě Low-Code jde primárně o usnadnění práce pro vývojáře, zkrácení doby vývoje, snížení nákladovosti celého vývoje, avšak jsou stále nutné technické znalosti odvětví. Na rozdíl od toho No-Code se snaží poskytovat řešení pro uživatele bez žádných zkušeností s vývojem. Obětí tohoto přístupu jsou ale flexibilita, technologické možnosti a možnosti customizace, spolu s dalšími problémy související s nemožností úpravy HTML, CSS, JS a dalších částí klasické webové aplikace. Každý z přístupů je tedy vhodnější pro jiné skupiny uživatelů, jiné aplikace a jiné cíle.

The essence of 'no code' is about empowering people to do things that previously were limited to specialists. Marketers are already using no-code tools across the full spectrum of marketing operations. (LXA 2022, sek. How marketers are using low code / no code)

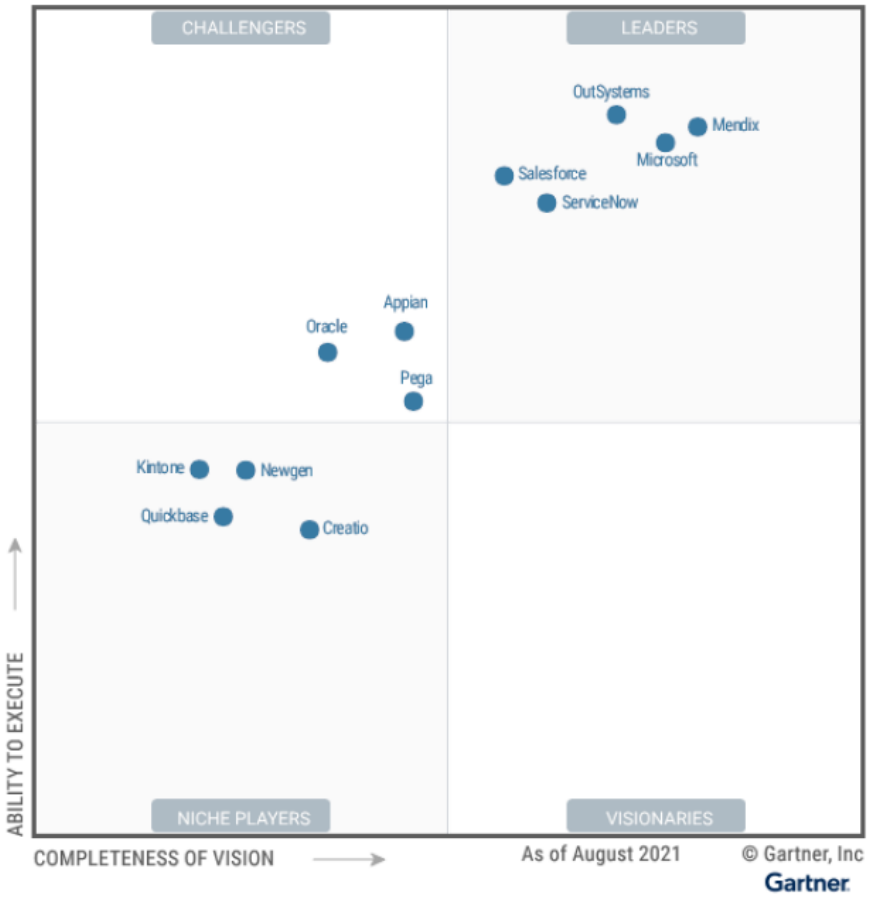
[33] Dle některých publikací jako (NoCode.Tech 2022) je možné ještě kromě No-Code odlišit i Zero-Code, ačkoliv tyto nástroje nepatří do oblasti programování. Dle autorů jde o programy jako Squarespace, MS Excel, MS Word apod.

4.1.2 Popis odvětví

Aktuální tržní hodnota Low-Code segmentu se odhaduje kolem 13 až 16 mld. USD s každoročním růstem 25% až 30% (BrandEssence a PR Newswire 2021) (Acumen Research and Consulting 2022). Je odhadováno, že k roku 2025 bude velikost trhu dosahovat 47,3 mld. USD a zároveň, dle publikace firmy Gartner (Wong et al. 2021), více jak 70 % firemních aplikací bude používat Low-Code, nebo No-Code technologie[34].

Jako důvody tohoto růstu jsou často zmiňovány probíhající digitalizace, přicházející Industry 4.0, přecházení firem do cloudu, nedostatek vývojářů a obecně IT specialistů a pandemie Covid-19 (Mearian 2022). Od Low-Code se v mnoha případech očekává příchod vyšší inovace a umožnění vývoje a poskytování self-service aplikací (Waldron 2022).

Jelikož se jedná o rapidně rostoucí a mladé odvětví, operuje v něm několik hlavních a poté mnoho specializovaných dodavatelů. Pro ilustraci je zde zobrazen Gartner magic quadrant ze srpna 2021, viz Obrázek 15, na kterém je vyobrazeno hned 12 dodavatelů, celkově jich je ale více jak 220 (Shuler 2022).



Obrázek 15: Gartner Magic Quadrant (Wong et al. 2021)

[34] Aktuálně to je méně než 25% (Wong et al. 2021).

Trh je možné segmentovat primárně pomocí use-case jednotlivých Low-Code platforem. Platformy, které již byly dříve v práci zmíněny, a které Gartner identifikoval jako tzv. Leadery, je možné popsat, jako univerzální platformy pro tvorbu jakýchkoliv firemních aplikací. Zároveň jde většinou o firmy s globální působností a často zaštítěné velkou matkou[35]. Dále ale existuje mnoho specializovaných platforem, jako je např. Creatio, která má mnoho zákazníků z bankovního, finančního nebo pojišťovnického sektoru, Kintone, která se specializuje na japonské a čínské manufakturní firmy, nebo Newgen, která se více orientuje na moderní AI/machine learning řešení či na tzv. microservices (Wong et al. 2021).

Další významnou segmentací trhu je dle velikosti cílových zákazníků, kde obvykle větší poskytovatelé, jako je třeba Mendix, cílí spíše na velké firmy a mají přímo pro ně postaveny své platební a hostovací modely, čehož opět jiní poskytovatelé využívají a specializují se naopak na firmy malé či střední, jako je např Kintone.

Poslední segmentaci, kterou bych chtěl zmínit je geografická segmentace. Firmy uvedené v Gartner magic quadrant na Obrázek 15 mají povětšinou globální charakter, avšak je nutné poznamenat, že existuje mnoho zemí, jako například Čína či Japonsko, kde se firmy a platformy často specializují jen na tuzemské zákazníky a podmínky (Interesse 2022). Existují i české platformy, jako je Jetveo či Intrexx (Jetveo 2022) (Intrexx 2022).

4.2 Platforma Mendix

V této kapitole bude specifikována platforma Mendix, a to primárně z funkčního a bezpečnostního hlediska. Je důležité názorně ukázat, jak platforma funguje pro účely dalších kapitol, primárně kapitoly 5, která obsahuje mnoho výpisů z Mendixu. Dále je důležité věnovat se bezpečnostním prvkům celé platformy pro účely vyhodnocování bezpečnosti autentizačních metod.

4.2.1 Popis firmy a platformy Mendix

Mendix jako právní entita vznikla v roce 2005, založena v Nizozemí s cílem změnit zažitá paradigma vývoje aplikací a „přemostit propast mezi byznysem a IT“. Propastí je obrazně znázorněn rozdíl a nepochopení dvou důležitých částí při vývoji SW, a její přemostění mělo být umožněno pomocí nových technologií, přístupů, nástrojů a metodiky (Mendix 2022v).

Dnes je jednou z hlavních firem v Low-Code odvětví, soustředící se na větší zákazníky, s největším úspěchem u firem z finančních a výrobních odvětví. V nedávné době firma přemístila své sídlo do USA, ale stále si zachovává významnou prezenci v Evropě, kde její aplikace využívají např. nizozemští státní dopravci i pošty či německá výrobní firma Continental.

[35] V případě Mendixu jde o firmu Siemens, u Power Apps jde o Microsoft, Salesforce mimo Low-Code podniká v mnoha jiných IT oblastech apod.

Jejími hlavními rozdíly, kterými se snaží diferenciovat se od konkurence je fúze byznysu a IT v rámci platformy, podpora více cloudového či on-premise implementaci a podpora nativních aplikací. Zároveň se jedná o velmi inovativní firmu, zaměřující se na nové trendy jako je např. IoT, a poskytující nová řešení pro stávající problémy, kde jako příklad je možné uvést nástroj na kontrolu kódu MxAssist Performance Bot[36], nebo Data Hub pro integraci dat (Wong et al. 2021).

4.2.2 Mendix Developer Portál

Každý projekt začíná v Mendix Developer Portálu. Jedná se o webové rozhraní, kde je možné zakládat a spravovat projekty, přistupovat k Mendix Marketplace, vstoupit do dokumentace, akademie, fóra a dalších částí celého Mendix ekosystému.

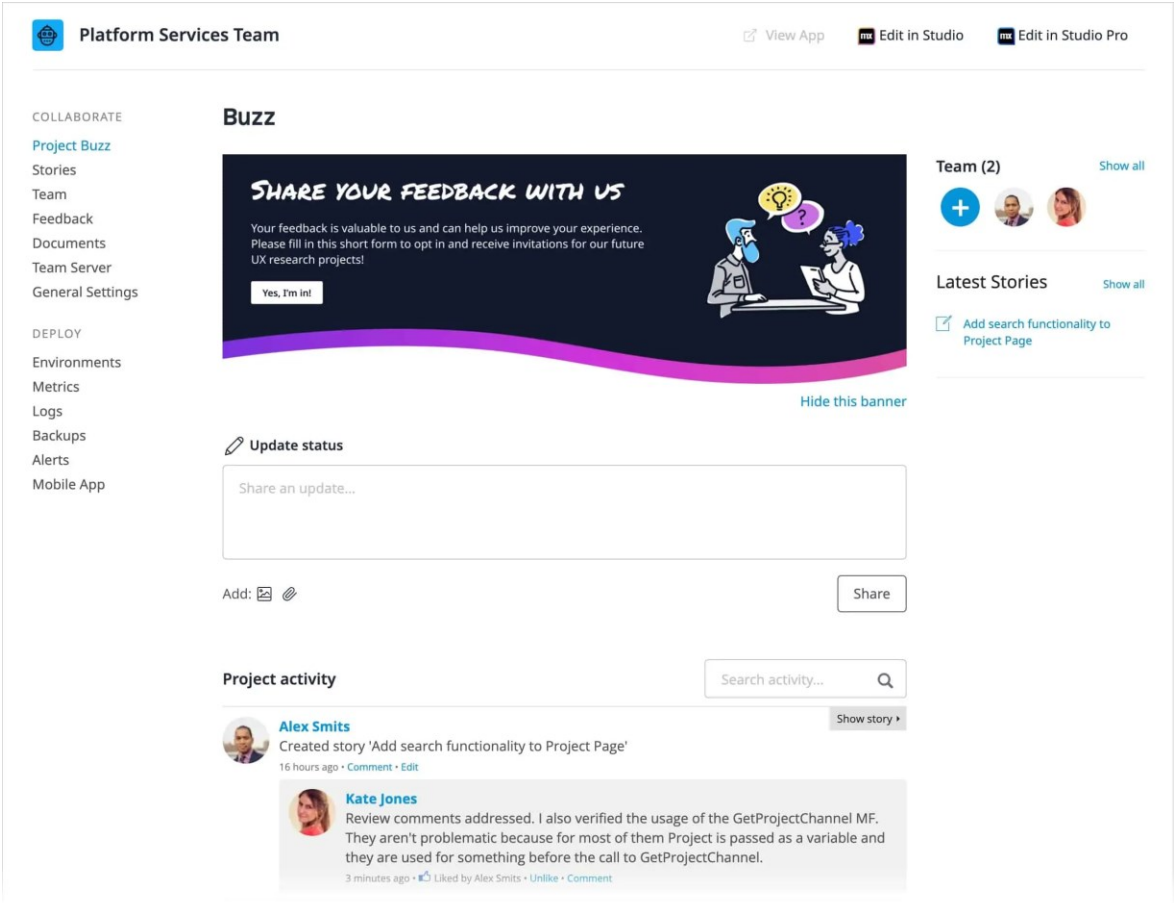
V tomto prostředí neprobíhá žádný vývoj, ale je velmi důležité pro kolaboraci a spolupráci se zákazníky. Zároveň se zde nastavuje, kdo má k jakému projektu přístup, jaká práva v rámci vývoje bude mít, a mnoho dalších projektových nastavení a přehledů, jako například:

- Přehledy
    - Chat
    - User Stories
    - Tým
    - Feedback
    - Dokumentový server
    - Týmový server – SVN/GIT
    - Prostředí
    - Logy
    - Zálohy
    - Metriky
- Nastavení
    - Role lidí v projektu
    - Nastavení jednotlivých prostředí
    - Nastavení cloudového node
    - Nastavení záloh

Obecně je portál využíván primárně manažery vývojářského týmu a zákazníkem, kteří zde společně domlouvají user stories, sledují/tvoří feedback, sledují postup vývoje či nějak jinak spravují projekt. Po dokončení projektu je primárním uživatelem správce prostředí či technický kontakt, který se zákazníkem spravuje chod aplikace. Samotní vývojáři se zde příliš nepohybují, jelikož se k mnoha položkám dostanou již z vývojářského IDE[37] a obvykle

[36] Jde o robotického pomocníka Low-Code vývojáře, který proaktivně kontroluje kód a analyzuje jej z modelového, architektonického a funkčního hlediska za účelem zvýšení výkonnosti a bezpečnosti.
[37] Vývojové prostředí. Eng. Integrated development environment

jej otevřou jen pro nastavení prostředí, pokud to je potřeba. Příklad úvodní stránky portálu na Obrázek 16.



Obrázek 16: Ukázka Mendix Developer Portálu (Mendix 2022e)

4.2.3 Komponenty

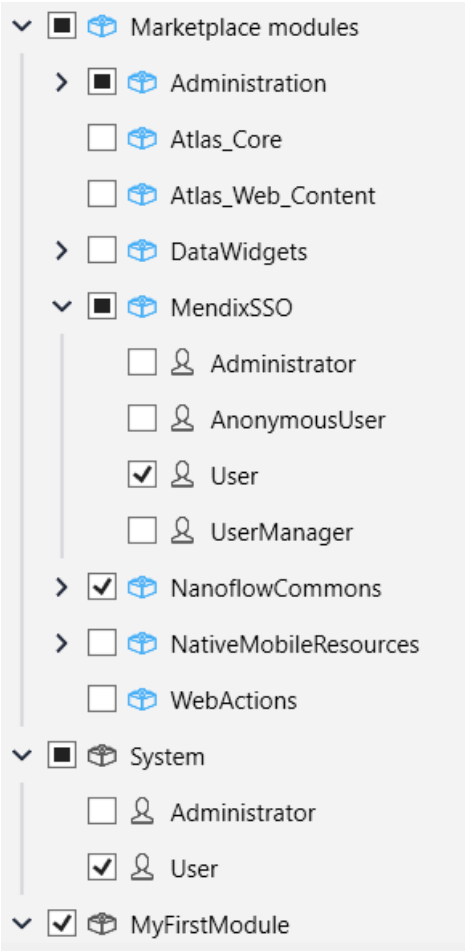
Programování v platformě Mendix probíhá primárně používáním vestavěných komponent, které programátor propojuje a skládá. V práci již byly uvedeny ukázky zápisu z Mendixu, viz Obrázek 11 a Obrázek 12, a v této kapitole budou jednotlivé prvky blíže představeny.

Projekt

Také někdy nazýván jako aplikace, projekt obaluje celou aplikaci a vytváří z ní jednotný celek. Obsahuje všechny systémová nastavení, jako je například umístění databáze, nastavení runtime, jazyková nastavení, certifikáty pro webové služby, či deployment nastavení. Zároveň přímo pod něj spadá celkové bezpečnostní nastavení, a to hlavně projektových rolí, hashovacího algoritmu a heslové politiky.

Jak fungují role v Mendixu bylo již částečně rozebráno v podkapitole 3.1.3, zde bych ještě chtěl doplnit, že každá projektová role se skládá z více modulových rolí. Je to proto, aby bylo umožněno snadné přenášení modulů, kvůli čemuž si každý modul definuje své bezpečnostní pravidla a na úrovni projektu dochází jen k jejich spojení do jedné velké uživatelské role, se

kterou poté pracují uživatelé v běžící aplikaci. Pro ilustraci je pod tímto odstavcem přiložen Obrázek 17, na kterém je možné vidět definici role User. Při vývoji aplikace mohu pomocí checkboxů jednoduše určit, jaké modulové role celková projektová role bude obsahovat.



Obrázek 17: Projektová role User (zdroj: Autor)

Důležitou součástí projektu je ještě navigace, ve které je možné definovat název a ikonu aplikace[38], společně s nastavením hlavního menu a home page.

Aby s projektem mohl programátor pracovat, musí si jej stáhnout na své lokální zařízení z projektového GIT nebo SVN serveru, který je hostován Mendixem.

Modul

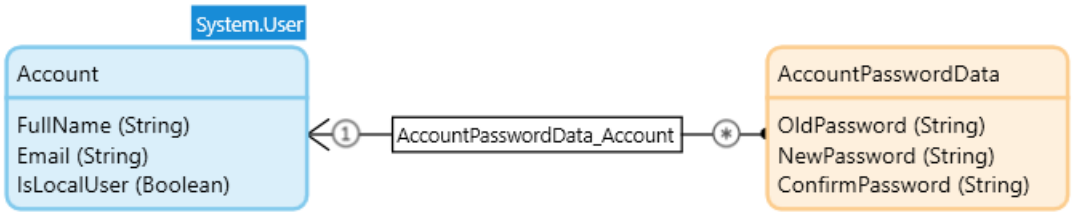
Moduly slouží k rozdělení projektu do jednotlivých celků. Toto rozdělení může být dle funkčnosti, účelu či původu modulu – proto existuje více typů modulů: systémový modul, UI Resources modul, marketplace modul a klasický modul.

Při tvorbě aplikace se obvykle pracuje se zhruba desítkami modulů, záleží ale ovšem na velikosti a požadované funkcionalitě. Nejvíce je přitom obvykle těch marketplace modulů,

[38] Tak, jak ji poté uvidí uživatel na záložce prohlížeče.

kterých je často více jak 60 %, a UI Resources modulů, kterých bývá tak 30 %. Systémový modul je vždy jen jeden, jelikož jde o specifický modul obsahující pevně dané entity a metody pro základní chod aplikace. Zbytek jsou klasické moduly, což jsou moduly, které nepochází z marketplace a byly vytvořeny programátory při vývoji.

V Mendixu obsahuje každý modul svůj doménový model a své vlastní Security nastavení



Obrázek 18: Ukázka doménového modelu (zdroj: Autor)

Doménový model je schéma databáze popisující entity, se kterými je možné pracovat, a jejich atributy. Na Obrázek 18 je znázorněn doménový model modulu Administrace, který obsahuje 2 entity: Account a AccountPasswordData. Každá entita má definované své atributy, jejich datové typy a omezení, asociace, generalizace, validační pravidla, handlery událostí, indexy, delete behaviour a perzistenci.

V rámci security je možné nastavit read/write access pro jednotlivé atributy a práva na create/delete operace v rámci modulových rolí. Stejně je možné nastavit práva spouštět či otvírat Microflows, Nanoflows a jednotlivé stránky.

Microflow

Tato a následující komponenty se již věnují přímo logice a vzhledu aplikace. Microflow, MF, je komponenta umožňující provádět logické operace. Jako příklad je možné uvést tvorba nových entit, načtení entit z databáze nebo otevření stránky. Při běhu aplikace se všechny microflowy vykonávají v runtime serveru, není tedy možné je využívat v off-line aplikacích a minimálně vytěžují klienta[39] (Mendix 2022p).

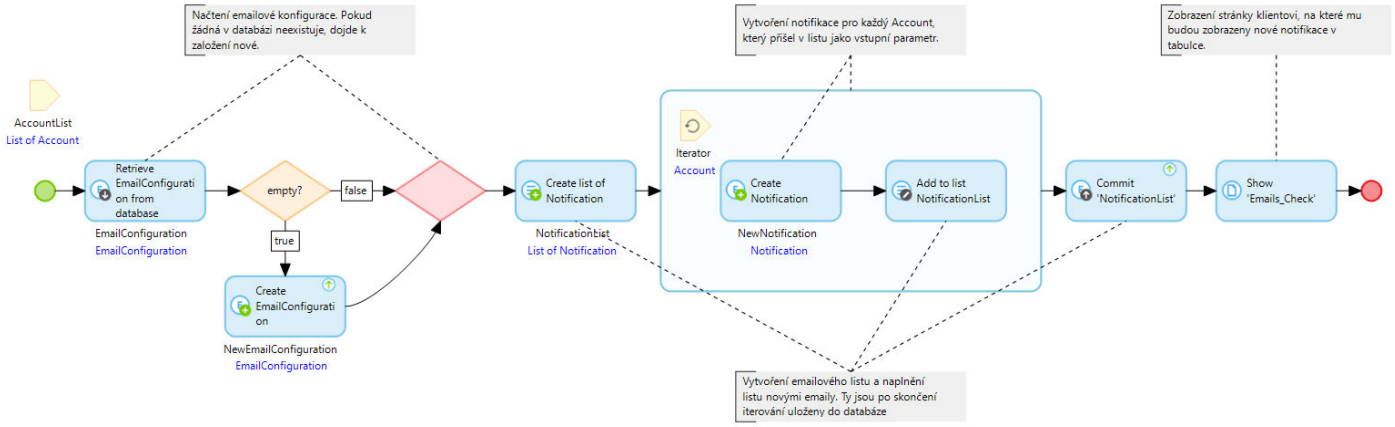
V Mendixu každý microflow začíná počáteční událostí, znázorněnou zelenou tečkou a končí v konečné události, červené tečce. Platí pravidlo, že počáteční událost je jen jedna, zatímco konečných může být i více.

Mezi těmito událostmi se nachází všechny aktivity, které microflow postupně ve směru propojujících šipek provede. Tyto aktivity mohou pracovat s objekty (klasicky CRUD operace), pracovat s listy objektů, volat jiné microflows nebo Java akce, provádět tzv. klientské operace, což může být například otevřít/zavřít stránku, stáhnout soubor, poslat zprávu, či synchronizovat data. V rámci notace se jedná o modré obdélníky.

[39] Tj. obvykle prohlížeč uživatele, který daný microflow spustil.

Někdy je nutné sled aktivit rozdělit na dva jiné směry, k tomu je možné použít tzv. decision. Tyto komponenty obsahují vstupní pole, kam je možné zapsat podmínku, dle které se bude rozhodovat výsledný chod microflowu. Značí se žlutým diamantem. Existuje poté ještě červený diamant, tzv. merge, který dokáže naopak více cest sloučit do jedné.

Posledními důležitými částmi microflows jsou cykly a parametry. Parametry jsou entity či hodnoty, které jsou poslány do microflow při jeho spuštění. Cykly existují v Mendixu 2 – tzv „For Each Loop“ a „While Loop“.



Obrázek 19: Okomentovaný vzorový microflow (zdroj: Autor)

Na Obrázek 19 je možné vidět microflow, který obsahuje většinu dříve zmíněných prvků. Na začátku se nalézá parametr „AccountList, a start event, po kterém následuje načtení dat z databáze a decision kontrolující, zdali se podařilo konfiguraci načíst. Pokud ne, dojde k vytvoření nové konfigurace, pokud již ale existuje, jde se přímo na vytvoření listu pro výsledné notifikace, které se poté reálně tvoří v iteracích pro každý Account, přítomný ve vstupním parametru. Následně se tyto notifikace uloží do databáze a uživateli se ukáže stránka „Emails_Check“.

Nanoflow

Nanoflowy, NF, jsou velmi podobné Microflowům, a to jak v účelu, tak i v notaci. Liší se avšak ve způsobu jejich vykonávání – Nanoflowy jsou prováděny na zařízení klienta a nespouští se v runtime serveru jako microflowy. Díky tomu mohou zároveň využívat funkcí jako je geolokace, bioautentizace, vytáčení hovorů, odesílání SMS a zároveň se hodí na pokročilé validace či spouštění JS kódu.

To má několik výhod: zaprvé není nutné namáhat server zbytečnými requesty, pokud je možné funkcionalitu provést již na straně klienta a zadruhé je možné tímto způsobem budovat offline aplikace. Avšak je zde limitace, že není možné komunikovat s databází, a proto není možné používat akce pro tvorbu nových objektů, jejich ukládání, načítání z databáze a rollbackování (Mendix 2022q).

Best practice při vývoji Mendix aplikací je proto kombinovat nanoflows a microflows dohromady tak, aby byly všechny úkony smysluplně a efektivně rozděleny mezi klienta

a runtime server. Tím dochází k optimalizaci zátěže, zlepšení responzivity, zkrácení prodlev mezi jednotlivými akcemi spolu se zvýšením uživatelského komfortu a UX.

Stránka

Poslední z komponent, které bych chtěl blíže představit je stránka (page). Tato komponenta slouží ke zobrazování všech informací uživateli, umožňuje manuální editaci a tvoření nových entit či volání MF a NF. Mendix funguje jako SPA[40], proto je vždy uživateli zobrazena jen jedna stránka, která je překreslována za pomocí JS při každé nutné změně.

V rámci Mendix aplikace je vždy definována jedna stránka jako home page, přičemž je možné specifikovat různé home pages pro jednotlivé role. Z této stránky se uživatel může orientovat pomocí menu, tlačítek, či jiných komponent, které mění aktuálně zobrazovanou stránku.

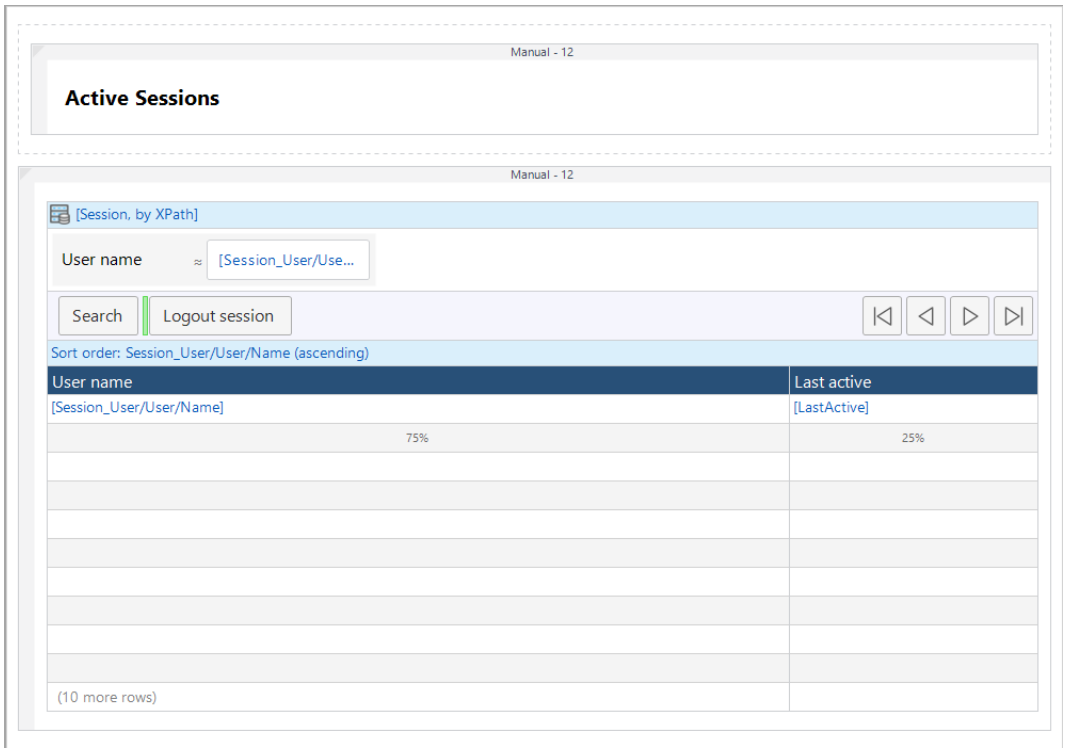
Obsah na stránkách se edituje pomocí tzv. datových kontejnerů (Data Containers) nebo widgetů. Datové kontejnery, jako je například data container, data grid, list view či template view, jsou prvky stránky, které jsou schopné načíst či pracovat s daty entit. Entity, které se mají zobrazit získají buď samy z Databáze, z MF, či kontextuálně[41]. Widgety jsou prvky, které mohou ale nemusí pracovat s daty entit, a jsou esenciálně všemi prvky, které jsou zobrazeny. Může jít například o text ve stránce, nadpisy, vstupní pole, obrázky, tlačítka, checkboxy, ale i o pokročilejší widgety jako celé mapy či grafy (Mendix 2022r).

Příklad stránky, na které je možné editovat data byl již v práci uveden jako Obrázek 11, pod tímto odstavcem je pro příklad ještě stránka, pracující s listem dat, viz Obrázek 20. Na vršku stránky je možné vidět statický nadpis (widget), pod kterým se nachází tabulka, zobrazující aktivní uživatele v aplikaci - tj. zobrazuje aktivní relace. Je možné všimnou si i konfiguraci tabulky, jako je například stránkování, filtr, řazení dat, zdroj dat, zobrazované atributy, či počet zobrazovaných řádků na stránku. Dále je v hlavičce tabulky tlačítko „logout session“, které umožňuje administrátorům ukončit vybrané relace.

Další vlastnosti stránky, které přímo netvoří její obsah, jako je tzv. title[42] či menu se definují pomocí rozložení a vlastností stránky. Stylování stránek je umožněno pomocí CSS či pomocí předdefinovaných CSS tříd, které může programátor při programování využívat. Pro pokročilejší stylování je možné použít ještě klasické HTML kontejnery či Bootstrapové Layout gridy.

---

[40] Single page application
[41] Kontextuálně znamená, že se použije entita, která je již načtená. Například v případě, že je v aplikaci tabulka s entitami, a uživatel by chtěl otevřít detail jedné z nich. Poté je možné do detail stránky poslat entitu kontextuálně z tabulky.
[42] Text, který se objeví v záložce browseru. Stejná funkcionalita jako HTML title element.



Obrázek 20: Příklad stránky s listem dat (zdroj: Autor)

4.2.4 Řízení bezpečnosti

Interní bezpečnost, jako jsou role, přístup k datům či jednotlivým komponentám nebo heslová politika, již byly v práci zmíněny dříve, tato podkapitola se avšak bude věnovat dalšími bezpečnostními opatřeními a procesy, které platforma a firma Mendix poskytuje.

Compliance a certifikace

Mendix splňuje následující bezpečnostní certifikace a standardy:

- ISO 22301 Certification
- ISO/IEC 27001 Certification
- ISO/IEC 27017 Certification
- ISO/IEC 27018 Certification
- NEN 7510 Certification
- ISAE 3000 Type II Assurance Report
- ISAE 3402 Type II Assurance Report
- SOC 1 Type II Assurance Report
- SOC 2 Type II Assurance Report
- SOC 3 Type II Assurance Report
- PCI DSS Level 1 Service Provider Attestation of Compliance
- HIPAA Assurance Letter
- Cyber Essentials (UK)

- CSA STAR Certification
- FSQS and FSQS-NL

Pro zajištění bezpečnosti podstupuje celá platforma alespoň jednou ročně analýzy rizik a bezpečnostní informační audity a provádí komplexní analýzy rizik, přičemž se soustředí hlavně na:

- organizaci & compliance
- bezpečnost platformy
- bezpečnostní modely
- runtime bezpečnost
- cloudovou bezpečnost

Zároveň se zaručuje, že všichni jeho zaměstnanci prošli vládně certifikovanou prověrkou, jsou vázáni striktním NDA[43] a také podstupují školení v oblasti bezpečnosti informací. Zaměstnanci, starající se o bezpečnost, jsou vedeni k certifikacím CISSP, CCSP, CIPP/E, CDPSE, či CISM (Mendix 2022s).

Platforma sama je postavena tak, aby dokázala zamezit nejčastějším útokům[44], jako jsou například SQL injections, cross-site scripting, CSRF a podobně, kde cílem je, aby vývojář mohl bez přílišného nastavování a implementování bezpečnostních prvků rychle tvořit aplikace. Proto je každá verze platformy vystavena penetračním a dalším testům, kterými se firma snaží eliminovat co nejvíce zranitelností.

Hashed String

Pro ukládání hesel se v platformě Mendix používá speciální datový typ hashed string. Při uložení hodnoty do tohoto pole se do databáze reálně uloží až osolený hash této hodnoty, kde pro tvorbu hashe je využit algoritmus definovaný v projektovém runtime nastavení. Je možné vybrat jeden z následujících algoritmů (Mendix 2022b):

- BCrypt – doporučený, je možné definovat BCrypt cost na škále 1-30
- SSHA256
- SHA256 – nedoporučuje se
- MD5 – nedoporučuje se

Hashed string je možné použít i v jiných situacích, avšak programátor je limitován faktem, že více než nastavit algoritmus hashovaní nemůže.

[43] Non disclosure agreement, dohoda o mlčenlivosti.
[44] Nejčastější útoky např. dle OWASP Top 10 či OWASP Low-Code/No-Code Security Risks (OWASP 2022a)

Šifrování

Všechna data se ukládají v šifrovaných databázích. Aplikační data se obyčejně nešifrují, avšak je možné z Mendix Marketplace stáhnout oficiální encryption modul, který nabízí funkcionalitu symetrického šifrování pro přenos či dodatečné šifrování dat. Stejně tak je možné si pro vyšší bezpečnost opatřit moduly na práci s cookies, či různými bezpečnostními algoritmy.

Soubory

Co se práce se soubory týče, zde je implementována minimální základní ochrana, kdy systém sám dokáže detekovat některé hrozby, jako jsou například tzv. zip bomby[45]. Zároveň je možné definovat white-list povolených souborových formát, ale pro kontrolu nahraných souborů na škodlivý obsah není dostupné žádné řešení. V aktuální době neexistují ani žádné moduly, které by tento problém řešily, a proto jediným řešením jsou externí API.

Logování

Logování je velmi důležitý prvek pro bezpečnost a správný vývoj aplikace. V rámci Mendixu existují 2 hlavní typy logování, a to aplikační a prostředí aplikace[46]. Logy v rámci aplikace jsou všechny logy, které jsou vytvořeny přímo v runtime aplikace za jejího chodu. Může jít například o logy přihlášení uživatele, logy o pokusu o přihlášení, či o logování vytvořené vývojáři. Druhý typ, tzv. logy prostředí, jsou čistě systémové záznamy, které není možné modifikovat či tvořit manuálně. Ty naopak zaznamenávají všechny operace, které provedli správci prostředí, tj. může jít například o spuštění/vypnutí aplikace, změna konfigurace prostředí, vytvoření/obnovení zálohy apod. V případě varovných či kritických událostí, se tyto logy zároveň mohou přímo emailově posílat technickému kontaktu aplikace.

Všechny logy se defaultně spravují centrálně na serveru Mendixu, odkud ale je možné je přenést na jiné umístění, např. během on-premise nasazení. Každý log obsahuje datum, typ, úroveň, text, node[47] a uživatele.

Vlastnictví dat a přístup k datům

Všechna data, ač na serverech Mendixu, patří zákazníkovi a přístup k nim mají zaměstnanci Mendixu jen s výhradním souhlasem zákazníka. Tímto se Mendix snaží zamezit tzv. Vendor Lock-In, kterého se mnoho firem obává při přechodu na Low-Codové či Cloudové platformy.

Po ukončení licence či využívání dat se Mendix zaručuje za bezpečné zničení dat, včetně záloh.

---

[45] Zip bomby jsou škodlivé soubory (obvykle archivy), které po otevření/rozbalení razantně narostou na velikosti, například milionkrát, čímž mohou zahltit systém, případně jej celý shodit.
[46] Autorský překlad. Originálně Application Logs a Environment Logs.
[47] Jde obvykle o jméno modulu. Slouží k usnadnění orientaci v log souborech a zlepšuje čitelnost.

Business Continuity a Disaster Recovery plan

V případě využití Mendix cloudu, je možné využívat vestavěné business continuity a disaster recovery procesy pro udržení aplikace v chodu. Architektura je postavená tak, aby v případě výpadku nějakého z runtime serverů či databází došlo automaticky co s možná nejmenší prodlevou k restartu a opětovnému nahrání dat. Pokud to je možné, všechny aktuální úlohy převezmou po dobu výpadku jiné servery a dojde ke kontaktování pověřených osob. Uživatelé by neměli pocítit žádný rozdíl a neměli by být omezeni ve využívání aplikace (Mendix 2022c).

Mendix provádí pravidelné testování těchto procesů každý kvartál, aby splnil náležitosti bezpečnostních certifikací, zvýšil bezpečnost a zabezpečil chod aplikací klientů.

5 Autentizace v prostředí Mendix

5.1 Webové metody autentizace

Následující podkapitoly se věnují webovým metodám autentizace, přesněji metodám:

- Přihlášení pomocí jména a hesla
- OTP,
- SSO,
- OIDC a OAuth,
- LDAP,
- A více faktorové autentizaci pomocí:
    - Google Authenticatoru,
    - Emailu

Použité moduly v rámci Mendixu jsou vypsány v kapitole 2.1. Jsou zde rozebrány vlastnosti a chování jednotlivých metod v prostředí Mendix Studio Pro 9.18.0, hodnocení a porovnání metod je obsahem následující kapitoly 6.

5.1.1 Jméno a heslo

Způsoby implementace

V rámci Mendixu existuje několik způsobů, jak autentizaci pomocí jména a hesla implementovat. Jedná se ovšem o rozdíly pouze v aplikační úrovni. Do databáze se vždy údaje uloží stejně. Možné způsoby implementace jsou:

1. Použití defaultní (HTML) login stránky
2. Použití vlastní Mendix stránky (či jiné komponenty) s defaultní login logikou
3. Použití vlastní Java akce

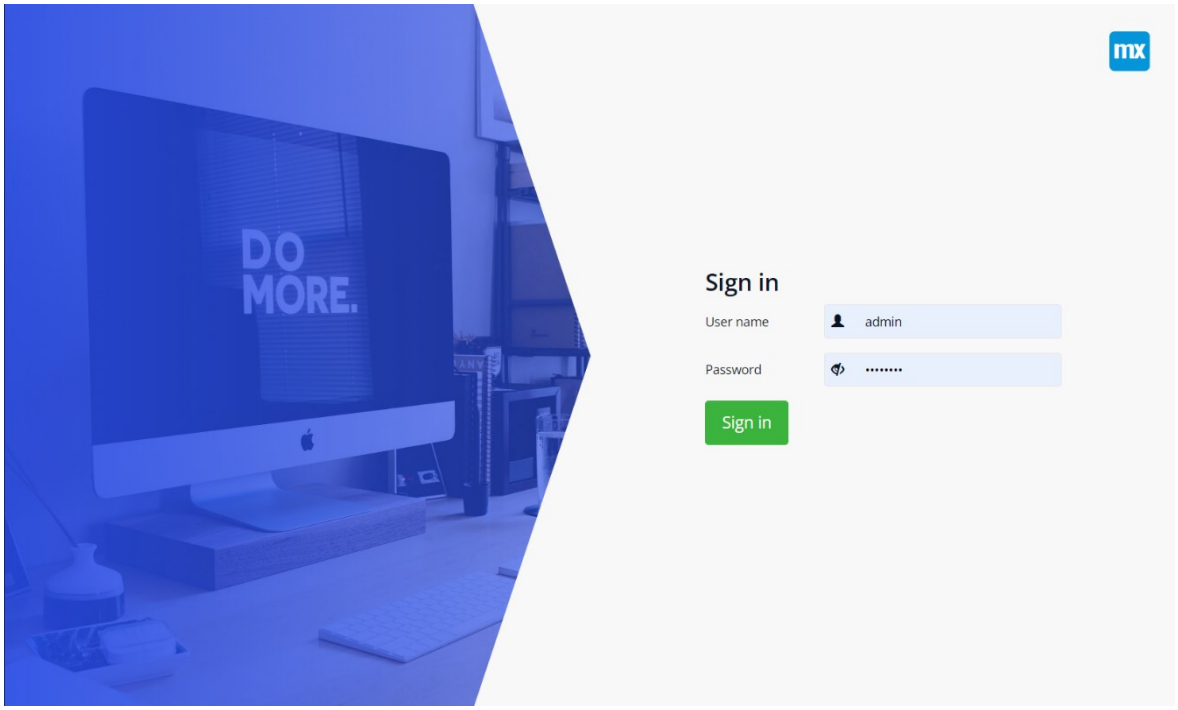
Použití defaultní (HTML) login stránky

Tento způsob je defaultní metodou přihlášení do každé Mendix aplikace[48]. Jedná se o jednoduchou předpřipravenou login stránku s formulářem, viz Obrázek 21, která obsahuje vstupní pole pro přihlašovací jméno a heslo spolu s tlačítkem pro odeslání formuláře, tedy přihlášení.

Formulář se odesílá jako standardní http POST request na /xas endpoint, což je speciální endpoint, který slouží ke komunikaci klienta s Mendix Runtime. Primárně slouží pro

[48] Za předpokladu že aplikace přihlášení vyžaduje.

zpracovávání CRUD operací, ale zároveň zpracovává i operace jako přihlašování. Poté, co byl login request na runtime zpracován, dojde buď k přihlášení uživatele nebo k zobrazení chybové hlášky uživateli.

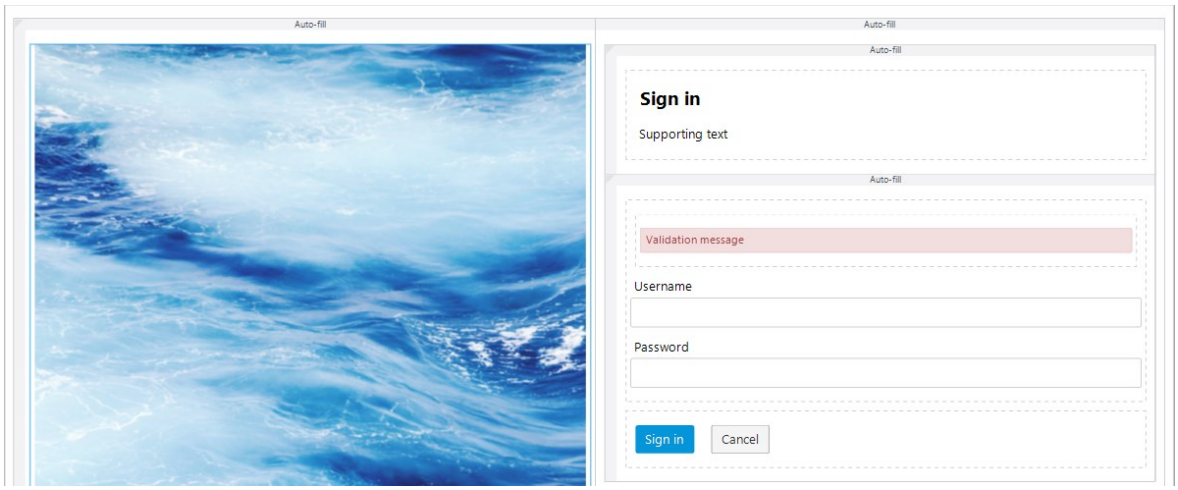


Obrázek 21: Defaultní login page (zdroj: Autor)

Omezením tohoto způsobu je primárně malá možnost customizace login stránky. Je sice možné stránku změnit úpravami HTML, CSS či JS kódu, ale není zde možné používat klasické Mendix komponenty, jako například widgety, MF či NF.

Použití vlastní Mendix stránky (či jiné komponenty) s defaultní login logikou

Dalším způsobem implementace této metody je využití vlastní login stránky. Oproti dřívějšímu způsobu je zde použita přímo Mendix stránka vytvořená v rámci projektu.



Obrázek 22: Vzor vlastní Mendix login stránky (zdroj: Autor)

V tomto případě je možné využít všechny komponenty, které Mendix nabízí a libovolně stránku nastylovat, doupravit a přizpůsobit dle funkčních a stylistických požadavků. Vývojář má zde mnohem větší volnost a možnosti.

Přihlášení probíhá stejně jako v případě defaultní stránky. Je nutné použít speciální input pole pro zadání loginu i hesla a zároveň je nutné použít i speciální „sign in button" a „validation message" pro vytvoření funkčního login formuláře, který se následně odesílá na /xas a zpracovává stejně, jako v případě minulého způsobu.

Nevýhodou tohoto způsobu je nutnost používání anonymní role, která musí být definována v projektu. Tuto roli automaticky obdrží nepřihlášení uživatelé, kteří vstoupí do aplikace a automaticky o ni přijdou po přihlášení. Toto samo o sobě nepředstavuje problém, avšak je zde prostor pro chyby, které mohou vést k bezpečnostním zranitelnostem a problémům. Zároveň je nutné stránku konfigurovat v rámci navigace projektu a defaultních domácích stránek.

Použití vlastní Java akce

Posledním způsobem implementace je vytvoření vlastní Java akce. Tento způsob je pochopitelně nejvíce možné customizovat pro potřeby aplikace, avšak je zároveň nejsložitější.

Pro použití této metody je nutné upravit after-startup MF[49] aplikace, ve kterém dojde k vytvoření tzv. „Login Action listeneru", který následně dokáže měnit výchozí chování loginu. Ukázka metody, provádějící přihlášení je znázorněna níže[50] ve Výpisu 1. Případně je také možné stáhnout si z Mendix Marketplace předpřipravené řešení pro vlastní login akci jako je např. modul „SignIn microflow for Mx7 / Mx8" (Appronto 2019).

Výpis 1: Login akce (Groot 2014)

```
public ISession executeAction() throws Exception
  {
    IUser user = Core.getUser(getContext(), this.userName);
    if (user == null)
         throw new AuthenticationRuntimeException("Login FAILED: unknown user '" +
this.userName + "'.");
    else if (user.isWebserviceUser())
         throw new AuthenticationRuntimeException("Login FAILED: client login attempt for
web service user '" + this.userName + "'.");
    else if (user.isAnonymous())
         throw new AuthenticationRuntimeException("Login FAILED: client login attempt for
guest user '" + this.userName + "'.");
    else if (user.isActive() == false)
```

[49] Jedná se o MF, který se spustí při spuštění aplikace.
[50] Ukázka metody je pro verzi Mx6. Verze, kterou se práce zabývá je Mx9.

```
        throw new AuthenticationRuntimeException("Login FAILED: user '" + this.userName +
"' is not active.");
    else if (user.isBlocked() == true)
        throw new AuthenticationRuntimeException("Login FAILED: user '" + this.userName +
"' is blocked.");
    else if (user.getUserRoleNames().isEmpty())
        throw new AuthenticationRuntimeException("Login FAILED: user '" + this.userName +
"' does not have any user roles.");

         // You can even call Microflows, like this:
    Map<String, Object> parameters = new HashMap<String, Object>();
    parameters.put("User", user.getMendixObject());
         //this will call the microflow LoginAllowed that should have a User parameter
and returns true if the user is allowed to log in
    boolean allowed = Core.execute(getContext(), "MyModule.LoginAllowed", parameters);
    if (!allowed)
        throw new AuthenticationRuntimeException("Login FAILED: user "+this.userName);

    if (!Core.authenticate(Core.createSystemContext(), user, this.password)) //password
check
        throw new AuthenticationRuntimeException("Login FAILED: invalid password for user
'" + user.getName() + "'.");

    ISession session = Core.initializeSession(user, this.currentSessionId);
    return session;
  }
```

Z pohledu přihlášení je nejdůležitější částí metody posledních několik řádků, kde pomocí metod Core.authenticate a Core.initializeSession dojde k autentizaci a vytvoření relace pro uživatele, který je tím přihlášen. Třída „Core“ (com.mendix.core.Core) je jednou z hlavních tříd Mendixu poskytující přístup k mnoha operacím v Mendix runtime serveru (Mendix 2023a).

Nevýhodou používání této akce je nahrazení původní login mechaniky, která mimo přihlášení poskytovala například i blokování uživatelů, kteří se opakovaně pokouší přihlásit špatným heslem a další funkcionality jako je např. logování. Při implementaci této metody je možné tyto funkcionality opět zprovoznit či doprogramovat. Dále je nutné mít přehled o tom, které moduly zasahují či upravují login akce, jelikož může snadno dojít ke konfliktům v této oblasti.

Samotné přihlášení se provádí jako u jiných způsobů implementace, tedy z defaultní či vlastní Mendix stránky.

Defaultní login

Ve všech způsobech implementace není přímo definován způsob, jak dojde k autentizaci uživatele. Ve všech případech dojde k volání /xas endpointu pomocí specializovaného formuláře buď z vlastní, či defaultní stránky. Ve všech případech dojde tedy k volání Mendix Runtime, který provede všechny nutné operace a vývojář má spíše omezené možnosti customizace[51].

Jak Mendix Runtime autentizuje uživatele či jak obecně provádí login se bohužel nepodařilo zjistit. Součástí dokumentace Mendix Studia Pro či třídy Core není přesný popis a po kontaktování přímo podpory Mendixu byla obdržena následující odpověď:

Unfortunately, this is not possible. The action behind the sign-in button is programmed into Studio Pro directly and is therefore not accessible (Mendix Support Team 2023, Osobní komunikace).

Uložení dat

Pro každou entitu uživatele[52] se ukládají následující atributy:

- Id
- Changeddate
- Active
- Createddate
- Lastlogin
- Webserviceuser
- **IsAnonymous**
- **Failedlogins**
- **Blocked**
- **BlockedSince**
- **Name**
- **Password**
- Submetaobjectname
- system$changedby
- system$owner

Přičemž pro účely autentizace je důležitých následujících 6: IsAnonymous, Failedlogins, Blocked, BlockedSince, Name a Password.

[51] Funkční customizace. V případě vizuální je možné použít všechny standardní Mendix komponenty či technologie používané při vývoji webových aplikací.
[52] Tj. entitu System.User.

IsAnonymous je boolean atribut, který definuje, zdali je daný uživatel anonymním či ne. Důležité pro autentizaci to je proto, že anonymní uživatelé používají speciální dočasné účty. Zároveň se často očekává přihlášení pouze od anonymních uživatelů.

Blocked (boolean), BlockedSince (datum) a FailedLogins (integer) jsou atributy řídící blokování uživatelů. Jak se tyto atributy používají je rozebráno v následující podkapitole.

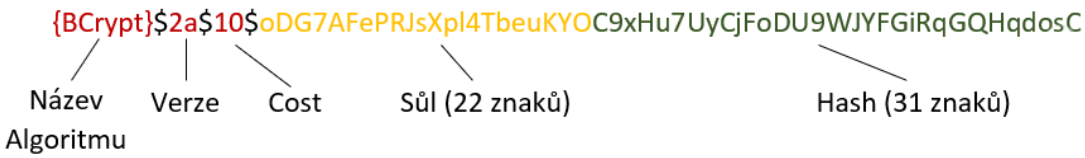
Poslední 2 důležité parametry jsou Name a Password. Atribut Name obsahuje přihlašovací jméno uživatele v plaintextové podobě. Password obsahuje hash hesla, jehož podoba je závislá na použitém algoritmu a jeho konfiguraci (viz 4.2.4). Příklad uložení jména a hesla je uveden níže[53].

- **Name**: Test
- **Password**:{BCrypt}$2a$10$oDG7AFePRJsXpl4TbeuKYOC9xHu7UyCjFoDU9W JYFGiRqGQHqdosC

Z uloženého hashe lze vyčíst několik faktů. Prvním je, že se jedná o Bcrypt. To je možné poznat přímo z prefixu {BCrypt} a následně poté i z identifikátoru algoritmu $2a$, který specifikuje že se jedná o Bcrypt[54], a charakter „a“ specifikuje jeho 2. verzi[55].

Dále, pomocí 10 v prefixu je možné určit, že cost BCrypt algoritmu při tvorbě hashe byla 10.

A finálně je ještě možné ze zbývajících znaků definovat sůl a hash, kde sůl tvoří 22 znaků a zbytek, 31 znaků, připadá na hash (PassLib 2020b). V těchto znacích jsou pomocí base64 zakódovány 128 bitové hodnoty soli a 192 bitové hodnoty hashe. Celková struktura vizuálně rozebrána viz Obrázek 23 níže.



Obrázek 23: Struktura uloženého hashe (zdroj: Autor)

Bezpečnostní opatření proti útokům

Mendix automaticky aplikuje ochranu proti brute force útokům při přihlašování uživatelů pomocí jména a hesla. Jelikož všechny login requesty jsou zpracovávány v Mendix runtime,

---

[53] Data získána přímo z Mendix cloud databáze pomocí stažení zálohy a následné analýzy pomocí pgAdmin4 (pgAdmin 2023) v6.18. Mendix aplikace používá algoritmus BCrypt s cost 10, jelikož to jsou defaultně přednastavené hodnoty.
[54] Pomocí identifikátoru algoritmu 2. Algoritmy mají své ID, které se přikládají k finálním řetězcům. MD5 je 1, BCrypt 2, NTHASH 3, SHA-256 4 atd. (PassLib 2020a).
[55] Verze seřazené vzestupně: 2, 2a, 2y a 2b.

není možné tuto ochranu obejít, ačkoliv je možné jako vývojář do této oblasti zasáhnout a funkcionalitu „ochromit“. To je ale relativně složité a musí jít o zásah do login mechanismu, tj. např při modifikaci samotné Java login akce.

Ochrana spočívá v blokování uživatelů, kteří se několikrát po sobě nedokázali přihlásit platným heslem. Ve výchozím nastavení jsou zablokováni uživatelé, kteří třikrát po sobě zadají špatné heslo, přičemž nezávisí na tom, jaký je časový rozestup mezi pokusy. Jakmile je uživatel zablokován, bude odblokován během následujících 5 minut[56]. Všechny neúspěšné pokusy a zablokování i odblokování uživatelů jsou logovány.

Pro blokování jsou používány atributy entity uživatele Failedlogins, Blocked a BlockedSince. Pokud Blocked=True, poté se uživatel nemůže přihlásit a musí počkat určitý čas, než jej systém[57], nebo v některých případech administrátor, odblokuje. FailedLogins slouží k počítání neúspěšných pokusů o přihlášení, které když přesáhnou určitou mezi, tj. obvykle 3, spustí proces na zablokování uživatele. BlockedSince uschovává datum a čas, kdy byl uživatel zablokován a je použit při automatickém odblokováním (Mendix 2023c).

Obvykle s těmito atributy pracuje pouze systém, ale je možné je upravovat i v rámci MF, což nabízí prostor pro vlastní logiku či úpravy této logiky. Nutné ale zmínit, že je důležité úpravy provést důkladně a následně řádně otestovat, jelikož je zde snadné udělat chyby. Např. pokud se nenastaví správně BlockedSince atribut, uživatel již nebude nikdy odblokován[58].

Další hrozbou, proti které Mendix již ve výchozím nastavení obsahuje protiopatření jsou injection útoky. Dle (Mendix 2022h) všechny komponenty, dostupné v Mendixu nativně, používají dostatečnou ochranu, jako např. parametrizované queries. Dále jsou všechny výstupy tzv. „Escaped“ a „Sanitized“, díky čemuž není možné provádět XSS útoky. Pokud by vývojář chtěl využít jiné, než mendixové komponenty, je doporučené, aby využil Mendix knihovny Community Commons (Mendix 2022f), která obsahuje dodatečné metody na ochranu vstupů a výstupů[59].

Doba Relace

Každá relace si, kromě jiných atributů, ukládá dva datumy řídící vypršení: CreatedDate a LastActive. CreatedDate obsahuje čas a datum, kdy byla relace vytvořena, tj. u anonymních uživatelů se bude jednat o přístup do aplikace a u neanonymních o přihlášení do aplikace[60]. LastActive je atribut, spravovaný v rámci runtime, který slouží k uchování času poslední aktivity klienta a k určení, zdali má být relace terminována či zachována.

---

[56] Jedná se o relativně novou změnu. Dříve, tj. u aplikací verze nižší, jak 9.11.0 fungovalo odblokování jinak. Uživatelé těchto verzí byli zablokování na dobu 0-5 minut, což bylo následně vyhodnocena jako chyba a opraveno při vydání 9.11.0 (Mendix 2022u).
[57] Systémová komponenta Cluster Manager. Viz podkapitola Doba Relace.
[58] To byl častý problém při příchodu verze 9.11.0, která jako první začala používat BlockedSince. Starší moduly s tímto atributem nepracovaly a nedocházelo poté k odblokování uživatelů.
[59] Knihovna implementuje metody obsaženy v OWASP Java HTML sanitizer (OWASP 2023).
[60] A zároveň se stejný čas vloží i do System.User LastLogin atributu.

Mendix následně řídí relace dle následující konfigurace (Mendix 2022a):

- **EnableKeepAlive** – Ve výchozím nastavení True. Určuje, jestli klient může provádět tzv. „KeepAlive“ requesty, které zabraňují automatickému ukončení relace. V případě nastavení na False bude uživatel v případě neaktivity odhlášen a relace ukončena po stanoveném čase SessionTimeout.
- **SessionTimeout** – Ve výchozím nastavení 10 minut. Zde je možné v sekundách nastavit čas, po kterém bude relace označena pro ukončení v případě neaktivity.
- **SessionKeepAliveUpdatesInterval** – Ve výchozím nastavení 100 sekund. Definuje, po jak dlouhém čase může být ukončená relace smazána z databáze.

Samotné ukončení relace provádí Cluster Manager. Jedná se o Runtime komponentu, která se mimo řízení relací zabývá blokováním uživatelů, obecně expirací procesů či objektů, synchronizacemi při nasazení a rušením nepoužívaných či nereferencovaných zdrojů (Mendix 2022d). Své akce vykonává periodicky, dle nastavení konfigurace ClusterManagerActionInterval, které je ve výchozím stavu nastavená na 5 minut.

Celkový proces automatického ukončení relace je ve shrnutí následovný (van der Hoek 2018):

- Pokud je uživatel v aplikaci a provádí akce, jako je načítání dat, volání MF, či jiné runtime operace, poté mu systém automaticky aktualizuje hodnotu LastActive. Pokud je EnableKeepAlive nastavené na True, pak tato hodnota je navíc aktualizována v pravidelných intervalech v čase SessionTimeout/2 sekund.
- V Runtime spouští každých 5 minut ClusterManager akce pro ukončení neaktivních relací. Relace se považuje za neaktivní, pokud od jejího času poslední aktivity uběhlo více jak 10 minut.
- Ukončené relace se po uplynutí SessionKeepAliveUpdatesInterval odstraní z databáze.

Heslová politika

Nastavení, které určuje heslovou politiku projektu, je možné najít v projektové bezpečnosti. Je možné zde nastavit pravidla, která se budou aplikovat při validování nových hesel před jejich uložením. Je možné nastavit 4 parametry: minimální délku, povinnost číslice, povinnost tzv. mixed case[61] a povinnost symbolu[62].

Ve výchozím nastavení je minimální délka 12 znaků, je povinné použít alespoň jednu číslici a mixed case. Symbol povinný není.

Jak již bylo zmíněno dříve, tyto pravidla se aplikují jen na nově tvořená hesla. Proto je možné mít stále v databázi hesla, která tuto politiku nesplňují, pokud byla vytvořena před zpřísněním heslové politiky. Kontrola probíhá v Runtime, pokud se uživatel pokusí změnit

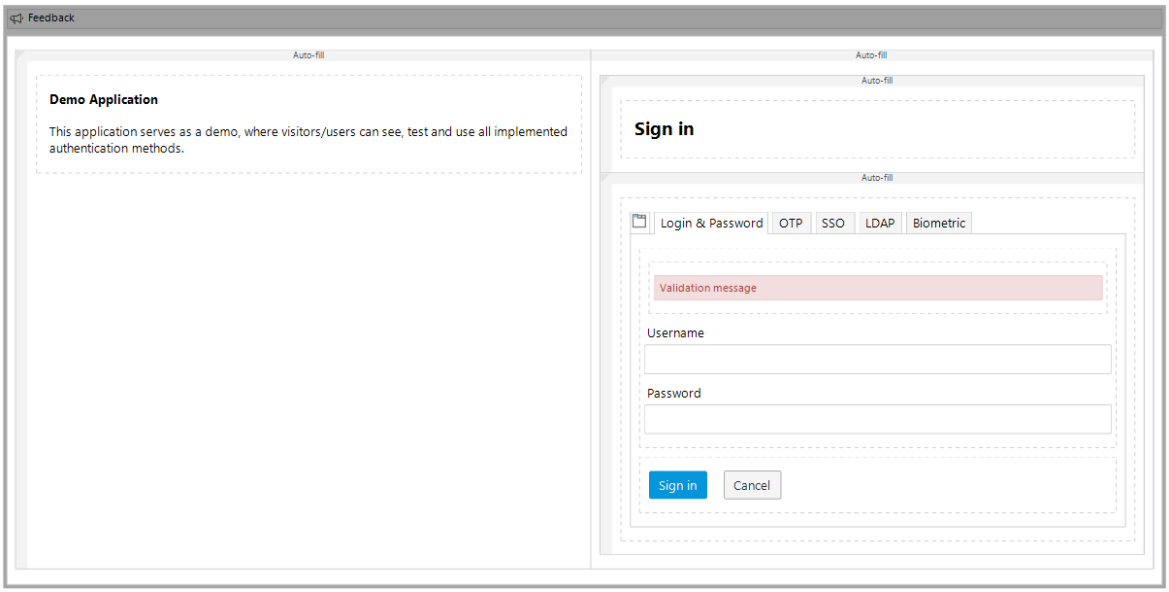
[61] Tj. povinnost použití jak velkých, tak malých písmen v hesle.
[62] Mezi symboly se řadí následující znaky: ` ~ ! @ # $ % ^ & * ( ) - _ = + [{] } \

či uložit objekt System.User či jiné uživatelské entity, které z něj dědí. Kontrola se provádí nad atributem System.User Password.

Implementace

Jelikož se jedná o nepříliš náročnou metodu, byla její implementace jednoduchá. V rámci demo aplikace byly implementovány všechny 3 dříve zmíněné metody. Jako hlavní bude ovšem použita metoda vlastní Mendix login stránky, a to primárně proto, aby bylo možné na tuto stránku přidávat další možné autentizační metody. Zbylé metody je možné najít v adresáři projektu.

Při tvorbě stránky byly použity standardní Mendix komponenty a musel být vytvořen nový page layout, jelikož je zde není vhodné zobrazovat nepřihlášeným uživatelům menu. Implementace znázorněna níže na Obrázek 24.



Obrázek 24: Implementovaná Login stránka (zdroj: Autor)

Pro změnu způsobu přihlášení je nutné upravit navigaci či konfiguraci projektu. Pro použití Java akce pro přihlášení je nutné v rámci AfterStartUp MF povolit Java akci „ReplaceLoginAction“. Pro použití výchozí login stránky je nutné zakázat anonymní uživatele v nastavení zabezpečení projektu.

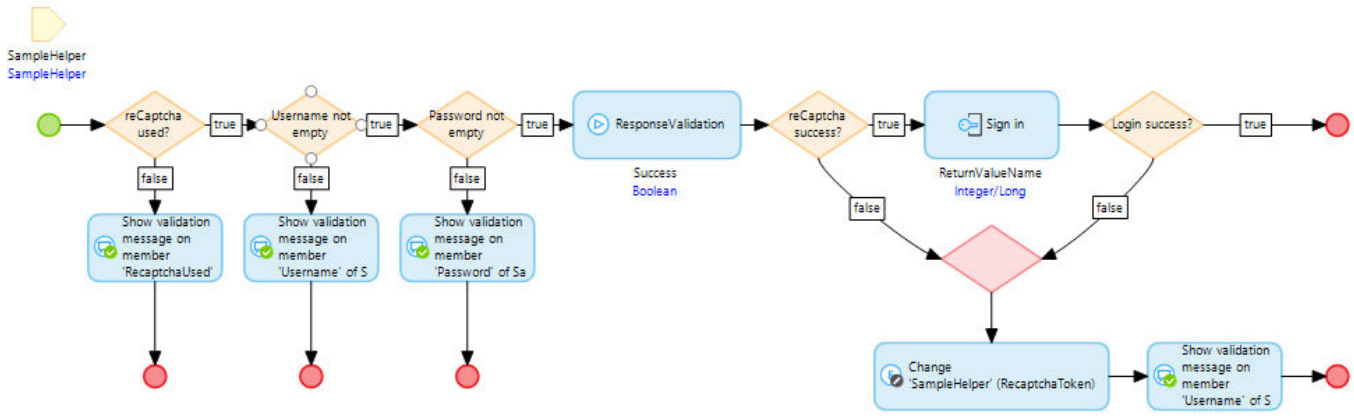
Proces registrace není nijak implementován a uživatele musí tvořit Administrátor přes své rozhraní v rámci User managementu.

CAPTCHA

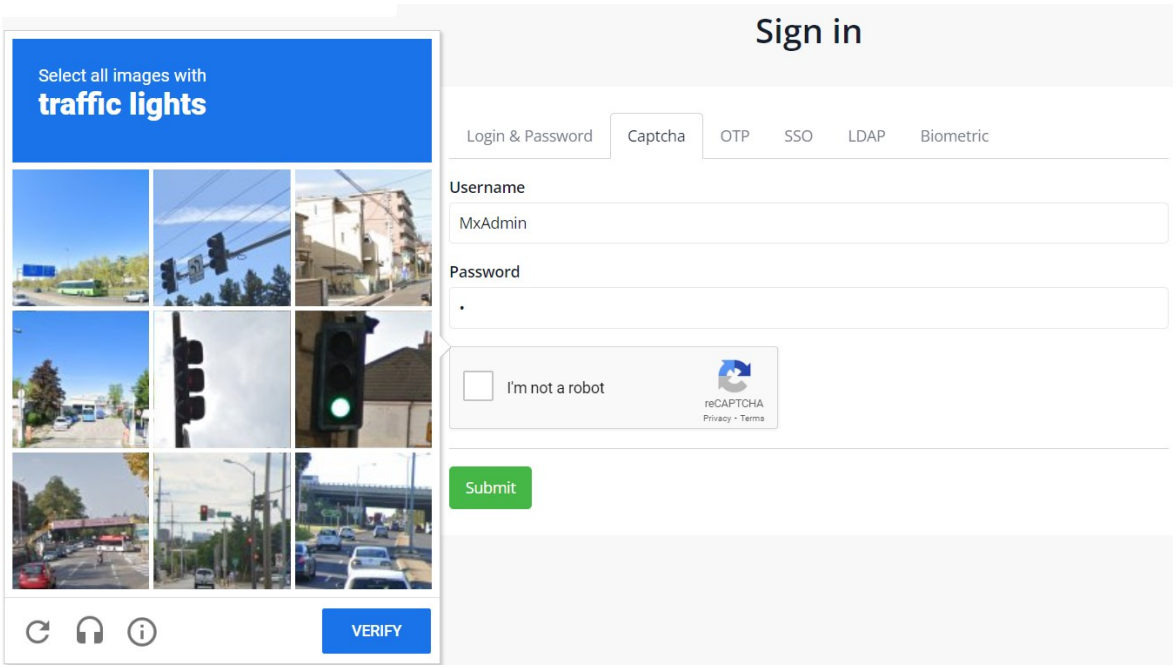
Následně byl implementován i modul pro zprovoznění CAPTCHA funkcionality. Problémem u implementace avšak bylo, že nebyl připraven pro verze Mx9 a výše. Prvně musel být modul stažen ve verzi Mx8, následně konvertován do Mx9 a finálně přenesen do projektu. Při konverzi nedošlo k žádným chybám či problémům, kromě několika stylistickým úpravám, které vycházejí ze rozdílů v UI Atlas modulech.

Samotná implementace nebyla obtížná, jelikož modul obsahuje dobrou dokumentaci, podle které šlo vše rychle zprovoznit. Aby modul fungoval, bylo ještě nutné vytvořit si Google reCaptcha SiteKey a SecretKey, které bylo nutné vyplnit do stejně pojmenovaných konstant v rámci modulu.

Posledním krokem byla již jen tvorba NF, který validuje vstupy od uživatele a provádí login, viz Obrázek 25. Výsledná podoba je zobrazená na Obrázek 26.



Obrázek 25: Login NF (zdroj: Autor)



Obrázek 26: Ukázka použití reCaptcha (zdroj: Autor)

5.1.2 OTP

OTP metoda, popsaná a vytvořená v této kapitole, slouží k autentizaci a přihlášení uživatele za pomoci časově omezeného jednorázového kódu. Je zde použita samostatně, tj. nefunguje na principu více faktorové autentizace, jelikož ta je rozebrána v podkapitole 5.1.7. Zároveň zde není použit žádný specializovaný modul, jak tomu je v dané podkapitole a celá metoda byla implementována autorem práce dle veřejně dostupných standardů, algoritmů

a dokumentace. V kapitole je popsán i průběh tvorby samotného modulu, avšak dále v rámci porovnávání se modul hodnotí jako by byl stažen z marketplace jako ostatní metody.

Limitace

Primárním omezením této metody je Mendix architektura. Ta nepracuje s autentizací pomocí OTP a naopak očekává, že každý uživatel se přihlašuje pomocí jména a hesla. Proto pro každého uživatele, který se úspěšně přihlásí za pomoci OTP, bude nutné vyřešit problém, jak jej do aplikace reálně přihlásit, jelikož každá implementace přihlášení[63] bude vyžadovat jméno a heslo.

Tato limitace se nakonec podařila překonat díky work-aroundu pomocí úpravy login Java akce (viz minulá kapitola). Řešení je popsáno níže v podkapitole Řešení limitace.

Koncept

Autorem byla navržena následující architektura OTP. Všechna OTP budou generována na serveru a následně posílána uživateli. Bude se jednat o sedmimístné číselné kódy, které budou uloženy na serveru a jejich platnost bude 10 minut. Účty, které se budou moci přihlásit pomocí OTP metody budou muset být specifikovány administrátorem, který je bude moci nastavit ve administraci uživatelů.

Konceptu posloupnosti činností vypadá následovně:

1. Administrátor
   1.1. Vytvoří/vybere účet a označí jej, jako účet používající OTP
   1.2. Systém pro tento účet vygeneruje náhodné heslo
   1.3. Administrátor sdělí Username uživateli
2. Uživatel
   2.1. Uživatel vstoupí do aplikace a zadá své Username
   2.2. Aplikace uživateli sdělí (pošle) OTP, který se zároveň uloží do databáze
   2.3. Uživatele získá OTP a zadá jej do Aplikace
   2.4. Aplikace zkontroluje platnost a správnost OTP. Pokud je vše v pořádku, aplikace uživatele přihlásí. Platnost OTP byla nastavena na 10 minut.

Poslední bod, tedy přihlášení uživatele je problematické. Jak bylo dříve zmíněno, pro přihlášení v Mendixu za použití defaultních komponent je nutné znát jméno a heslo. Proto bylo nutné specificky upravit login akci tak, aby dokázala pracovat s OTP kódy vybraných uživatelů.

Řešení limitace

Pro řešení problému byly analyzovány různé dostupné autentizační moduly a jejich vlastní Java Akce a způsoby, jak přihlašují uživatele. Např. SSO modul zmíněný v 2.1. Následně

[63] Skrze Java akci či NF sign-in akci.

dokumentace jako (Mendix 2023a) a jiné externí materiály jako např. Mendix fórum (Mendix 2023f). Výstupem z autorovy analýzy bylo, že metodou, která plní to, co je zde požadováno, je Core.initializeSession(). Jedná se o metodu, se kterou bylo možné se setkat i dříve v kapitole 5.1.1 a která je odpovědná za tvorbu a asociaci session uživateli.

Následně byla vytvořena jednoduchá Java akce, která měla za cíl validovat OTP kód uživatele a následně jej přihlásit pomocí Core.initializeSession(). Tato akce byla použita zkušebně v různých NF i MF, ale vždy došlo k chybě, že klient není autorizován, viz výpis níže:

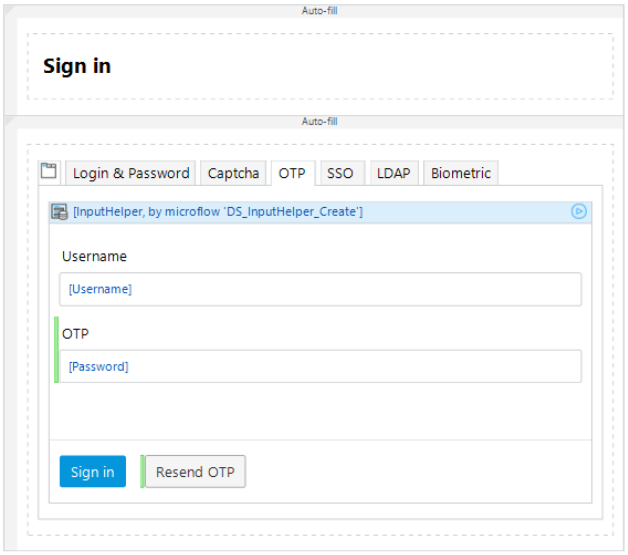
```
Client_Auth (info): The client is not authorized. Request failed with status code 419.
Removing token and session.
```

Autorský názor je, že kód byl v pořádku, ale problém nastával jeho umístěním do klientských akcí, které nebyly pro tyto aktivity autorizovány. Proto byl umístěn do custom Java login akce, viz minulá kapitola, a zde již fungoval. Problém byl tedy vyřešen, avšak vyvstal zde problém nový, a to, že úpravou login akce nebylo možné se přihlásit klasicky pomocí jména a hesla, jelikož nově zde systém očekával vstup OTP. Pro řešení tohoto problému byla akce rozšířena, aby dokázala pokrýt obě možnosti. Aktuálně tedy je možné se přihlásit jak pomocí OTP tak pomocí jména a hesla, kde validace provádí stejná akce a rozlišení je zařízeno pomocí toho, zdali aktuálně přihlašovaný účet používá OTP či nikoliv, tj. dle dřívějšího nastavení administrátora.

Obecně je při implementaci OTP tímto způsobem v Mendixu důležitější znalost Javy a Mendix Java knihoven, nežli samotného Mendixu. Implementovat stejnou funkcionalitu za pomoci nativně dostupných nástrojů není možné. V projektu se to podařilo částečně zprovoznit tak, že by systém vygeneroval uživateli nové heslo při každém přihlášení, uložil jej do uživatelovy entity a následně jej rovnou přihlásil pod tímto heslem. Pro uživatele na první pohled nejde poznat rozdíl, ale jedná se o mnohem méně bezpečnou metodu spolu s neefektivním ukládáním stále nových hesel, která si stejně mohl uživatel získat např. z Chrome DevTools.

Implementace

V případě OTP se jednalo o složitější implementaci jak v případě přihlašování jménem a heslem. Login page byla vytvořena podobně jako u předchozí metody, viz Obrázek 27. Jedná se o jednoduchou stránku s dvěma vstupními poli a dvěma tlačítky. Uživatel při vstupu na stránku vidí jen pole pro Username a tlačítko sign in, pokud pole správně vyplní a pokusí se přihlásit, bude mu odeslán OTP a zároveň se mu zobrazí pole pro OTP a tlačítko na opakované zaslání OTP.



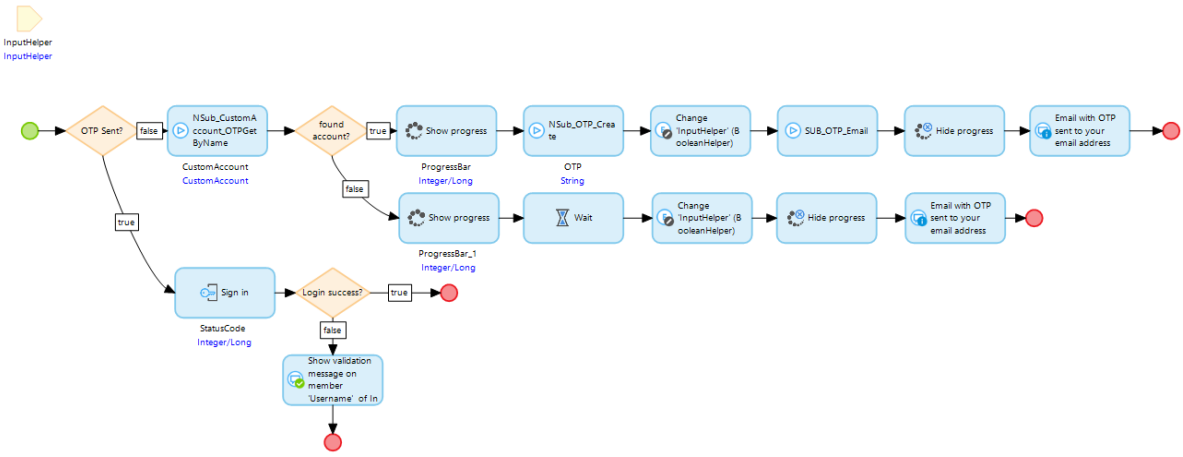
Obrázek 27: OTP Login stránka (zdroj: Autor)

Většina logiky v OTP mechanismu je obsažena v následujícím nanoflow, viz Obrázek 28. Je možné si jej rozdělit na 2 hlavní větve. První, vrchní, větev se vykonává pokud má dojít k odeslání OTP kódu. Je zde tedy validováno vložené Username, generován OTP kód a vytvořená entita uložena do databáze, kde je zároveň asociována s entitou uživatele. Před uložení kódu dochází ještě k jeho šifrování. Finálně je OTP kód odesílán emailem uživateli[64]. V případě, že není nalezen uživatelský účet, dojde k mimikování chování, jako by nalezen byl[65].

Druhá větev se vykonává, pokud má dojít k přihlášení. Je zde volána pouze „Sign in“ akce, jelikož ta následně spustí custom Java akci.

Poslední částí implementace bylo vytvoření administrátorského rozhraní pro OTP, kde může administrátor povolit či zakázat OTP přihlašování u uživatele a zároveň mít přehled o odeslaných OTP kódech a jejich platnosti.

[64] K odesílání emailů musel být stažen a nakonfigurován emailový modul Email Connector, (Mendix 2023b).
[65] Přihlašujícímu uživateli se zobrazí na náhodně stanovenou dobu od 2 do 3 sekund progress bar a následně se mu ukáže oznámení o odeslání OTP, ačkoliv k žádnému reálně nedošlo.



Obrázek 28: Primární OTP NF (zdroj: Autor)

Pro použití OTP metody musí být povolena Java akce „ReplaceLoginActionWithOTP“ ve AfterStartUp microflow projektu.

Doba Relace

Metoda pracuje se stejnou dobou relace, jak kapitola 5.1.1.

Bezpečnostní opatření proti útokům

Kvůli používaní kódů na jedno použití je metoda v určitých ohledech bezpečnější jak přihlašování jménem a heslem, viz 3.3.2. V rámci modifikace login akce bylo nutné v rámci implementace doplnit funkci blokování uživatelů, a to jak uživatelů přihlašující se pomocí OTP, tak i přihlašujících se pomocí jména a hesla.

5.1.3 SSO

V rámci práce byly zpracovány 2 různé SSO moduly. V obou případech se jedná o SSO řešení postavené na OIDC, avšak výrazně se liší ve svém doporučeném use case a způsobu práce s uživateli.

První modul, zpracovaný v podkapitole Mendix SSO, pracuje výhradně s Mendix účty, zatímco druhý, v podkapitole OIDC SSO, pracuje klasicky jako plnohodnotné SSO, které se dokáže připojit k různým IdP.

5.1.4 Mendix SSO

Jedná se o oficiální Mendix modul, který velmi specificky pracuje pouze s účty Mendixu. Není zde možné nastavit jiného IdP a zároveň jej není možné nakonfigurovat pro aplikace, které neběží na Mendix Cloud. Zatím se jedná o externí modul, který se musí do projektu dodatečně stáhnout, v budoucnu je ale možné očekávat, že bude přímo předkonfigurován a zasazen do všech projektových šablon Mendixu (Mendix 2022o). Již dnes je možné najít v projektech předpřipravené SSO login stránky, které slouží k zpřístupnění SSO. Souběžně

Mendix vyvíjí novou „build your own IdP, BYOIDP“ funkcionalitu, která by měla jít s tímto SSO následně integrovat (Mendix 2022t).

Komunikace

Komunikace probíhá standardně dle OIDC SSO, viz podkapitola 3.3.4, přičemž jako OpenID provider vystupuje přímo server Mendixu. Veškerá komunikace probíhá pomocí HTTPS protokolu.

Při pokusu o přihlášení je uživatel přesměrován na adresu aplikace s dodatkem „openid/login“, který je zpracován Mendix ILogin handlerem[66]. Při prvním pokusu o přihlášení bude uživatel přesměrován na Mendix, aby se přihlásil a následně autorizoval aplikaci, aby mohla využívat jeho data z profilu. Po úspěšné autorizaci přesměruje Mendix uživatele zpět do aplikace, kde bude request zpracován pomocí callback handleru[67]. Při všech dalších přihlášení již nemusí autorizovat aplikaci a je rovnou přihlášen.

Při odhlášení z aplikace dojde ke smazání všech uživatelových OpenID tokenů v aplikaci. Nedojde avšak k odhlášení z Mendixu. Pokud se uživatel pokusí následně opětovně přihlásit, bude tedy okamžitě přihlášen, a nemusí zadávat heslo či se jinak ověřovat[68]. Toto je správné chování Mendix SSO modulu, což bylo ověřeno u Mendix podpory.

Při opačné situaci, tj. při odhlášení z Mendix portálu, je uživatel odhlášen ze všech aplikacích, ke kterým byl přihlášen. K tomuto odhlášení dojde po provedení jakékoliv akce.

Tokeny

Mendix SSO funguje na stejném principu jako OIDC SSO, proto používá i stejné tokeny. Níže, viz Tabulka 7, je zobrazen částečný export databáze, kde je možné vidět základní informace o všech tokenech. Jedná se o algoritmem RS256 podepsané JWT zakódované v base64.

Doba expirace v prvním sloupci je uvedena v sekundách, platí tedy, že výchozí doba expirace access tokenu je 1 hodina, ID tokenu 1 den a refresh tokenu 30 dní. Tyto hodnoty se mohou změnit v konfiguraci modulu.

Tabulka 7: MendixSSO Tokeny (zdroj: Autor)

Expires In	Token Type	Value
3600	ACCESS_TOKEN	{AES2}W7MQvXwPhYl2hQ1E;3YzdCdK[...]
2592000	REFRESH_TOKEN	{AES2}0L7bCJyhwyNVnXA0;l1ZwAz3[...]

[66] Součástí ILoginHandler.java souboru modulu
[67] Součástí DefaultLoginHandler.java souboru modulu.
[68] Pokud se stále přihlášený u Mendixu a pokud je aplikace stále autorizovaná k využívaní jeho dat.

86400	ID_TOKEN	{AES2}l/aH1kKwR309gWoQ;Hphgs82[...]

Bezpečnost

Při ukládání tokenů v databázi jsou všechny tokeny šifrovány pomocí AES2, kdy klíč je tvořen samotným systémem při nasazení aplikace do cloudu. Vývojář jej nemůže přenastavit. Access token a refresh token jsou navíc šifrovány pomocí A128GCM.

Při tvorbě uživatelských účtů v aplikaci se pro nové účty generuje náhodné 48 místné heslo. Uživatelské jméno je nastaveno jako UID z Mendix IdP. Při každém přihlášení se uživatelská entita aktualizuje, a všechny atributy se nahradí novými dle nastavení. Toto nastavení je možné částečně měnit ve specializovaných MF.

Dále je nutné pro správnou funkčnost modulu adekvátně nastavit projektovou bezpečnost. MendixSSO pracuje se 4 modulovými rolemi: Administrator, UserManager, User a Anonymous. Je nutné každé projektové roli, která má SSO využívat některou z těchto rolí nastavit.

Implementace

Jelikož se jedná o silně předkonfigurovaný modul, nebyla implementace nijak složitá. Do projektu byl stažen modul, a byly pouze upraveny 2 MF, MendixSSO_CreateUser a MendixSSO_UpdateUser, kde se pouze změnila entita uživatele na entitu používanou v aplikaci. Následně bylo upraveno administrátorské menu, aby bylo možné mít přehled o tokenech.

Posledním krokem implementace bylo přenést login funkcionalitu z původní stránky „login-with-mendixsso-button.html“ do vlastní login page, čehož bylo dosaženo jednoduchým NF.

Tím byla zprovozněna celá funkcionalita v aplikaci a dále bylo jen nutné nastavit přístupy uživatelů, viz další kapitola. Mendix SSO poskytuje ještě možnosti další konfigurace v rámci konstant, které jsou v projektu. Jedná se primárně o textace, algoritmus pro šifrování a doby expirací. Všechny ostatní konfigurace jsou provedeny samy Mendixem při nasazení na Mendix Cloud.

Doba Relace

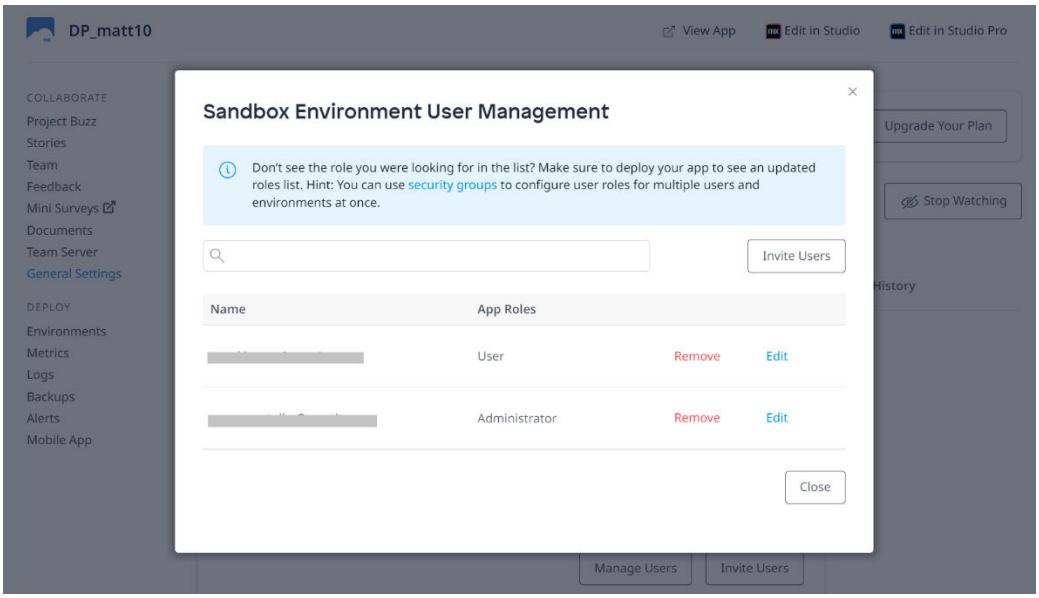
Doba relace vyplývá z doby platností tokenu, viz kapitola výše, a nastavení aplikace. Dle autorova názoru je doba platnosti tokenů nastavená příliš dlouho. Refresh token je nastaven na 30 dní a při každém odhlášení a přihlášení jsou tokeny smazány a opět vytvořeny.

Dále je problematické, že se není možné odhlásit od IdP při odhlášení z aplikace. Odhlášení sice zničí relaci v aplikaci, ale stále existuje relace u IdP, díky které je možné se do aplikace přihlásit opětovně bez zadání hesla. Podobný problém nastává i při vypršení tokenů, kdy není implementována žádná kontrola jejich platnosti jinde než při přihlášení, a proto je možné být přihlášen a pracovat s aplikací déle.

Správa uživatelů

SSO uživatele spravují členové projektu, kteří mají na tuto funkci práva. Tj. při defaultním nastavení se jedná o Scrum Mastery. Tito členové mohou v portálu projektu v sekci General Settings v rámci Access Management spravovat uživatele pro jednotlivá prostředí. Správou je myšleno přidávat, mazat a měnit uživatelské role. Ukázka menu pro nastavení uživatelů je níže na Obrázek 29.

Aby mohl být někdo přidán, jako SSO uživatel, je nutné, aby měl vlastní Mendix účet. To znamená, že nemusí být členem projektu, tj. ani mít přístup do samotného portálu. Jakmile je uživatel pozván, přijde mu email s odkazem a informacemi o pozvání.



Obrázek 29: Management SSO uživatelů (zdroj: Autor)

Limitace a omezení

Pro tento modul jsou zásadní následující limitace a omezení:

- Nemožnost využití jiného IdP než Mendixu
- Nemožnost využití mimo Mendix Cloud
- Aktuálně se modul často aktualizuje a rapidně mění
- Není možné testovat lokálně
- Modul není možné využít v nativních aplikacích

5.1.5 OIDC SSO

Pro zprovoznění této metody byl použit oficiální Mendix modul pro OIDC SSO. Oproti předchozímu modulu je zde poskytována volnost ve výběru IdP a obecně vyšší možnost konfigurace jednotlivých prvků metody. Zároveň poskytuje možnost využití vícero IdP a již pracuje s předdefinovanými, často používanými IdP, jako je např Salesforce či Azure.

IdP

Pro testování aplikace a implementaci byl použit OpenID Connect společnosti Google, kde bylo možné si snadno a zařídit neplacené testovací prostředí (Google 2023). Tento IdP byl vybrán z několika důvodů, prvním je, že se jedná o populárního a dosti využívaného OIDC providera. Dalšími důvody je pak jednoduchost, rychlost a zdokumentovanost celého řešení, spolu s lehkým přístupem za pomoci standardních @gmail.com účtů.

Pro zprovoznění služby bylo nutné si vytvořit projekt v rámci Google Cloud a v něm zaregistrovat a nastavit nové prostředí. Musel být nastaveny povolené domény[69] a tzv. redirect URIs[70]. Po dokončení nastavení byly získány údaje pro registrace aplikace ke Google OIDC, jako je ClientID, ClientSecret a .well-known/openid-configuration[71]. Tyto údaje byly následně využity v nastavení modulu v demo aplikaci, viz podkapitola implementace a komunikace.

Správa uživatelů

Je prováděna na straně IdP. V aktuální implementaci s Google OIDC je možné použít jakýkoliv „@gmail.com“ účet pro autentizaci a přístup do aplikace. Je ovšem možné ale tyto uživatele restriktovat dle určitých podmínek, či přímo definovat tzv. whitelist.

Komunikace

Veškerá komunikace mezi aplikací a IdP je zprostředkována HTTPS. Při přihlášení je uživatel přesměrován na autorizační endpoint, kde se může přihlásit a získá potřebné údaje pro volání token endpointu a userinfo endpointu. Všechny tyto endpointy musí být nakonfigurovány v nastavení modulu.

Jaké atributy o uživateli jsou komunikovány je možné standardně upravit pomocí nastavení Scopes a Claims.

Tokeny

Stejně jako u dřívější metody byl proveden export databáze tokenů. Zásadní změnou oproti dřívější metodě je, že tokeny se v rámci OIDC SSO modulu ukládají do jedné entity. V tomto případě se také jedná o podepsané, algoritmem RS256, JWT zakódované v base64, avšak modul je schopný pracovat i s jinými algoritmy.

---

[69] Zde byla nastavena hodnota „mxapps.io“, jelikož na této doméně běží demo aplikace.
[70] Redirect URI je URI, kam je uživatel přesměrován po úspěšné autentizaci u IdP. Dle dokumentace modulu nastaveno „https://dpmatt10-sandbox.mxapps.io/oauth/v2/callback“.
[71] Jedná se o veřejnou URL, kde poskytovatelé poskytují informace o službě. Obvykle se jedná o standardizovaný JSON s údaji o endpoints, scopes a claims. Příklad u OIDC Google je https://accounts.google.com/.well-known/openid-configuration.

Tabulka 8: OIDC SSO Tokeny (zdroj: Autor)

Access Token	Refresh Token	Id Token	Expires in	User
ya29.a0AVvZVspduDDz4[...]	-	eyJhbGciOiJSUzI1NiIs[...]	0	111717868212527687 46

Oproti dřívější SSO se tokeny nešifrují při ukládání do databáze a ukládají se v takové podobě, v jaké přichází od IdP.

Bezpečnost

Každému uživateli, který se do aplikace registruje pomocí OIDC SSO[72], je nastaveno jméno jako atribut „sub“ z ID Tokenu. Jako heslo je nastaveno 16 místné heslo. Při každém přihlášení se aktualizují uživatelovy údaje.

Oproti minulé metodě je zde mnohem vyšší volnost customizace a úprav chování metody. Kromě možnosti výběru a nastavení služby IdP je zde možné jakkoliv upravit MF řídící celý autentizační proces či Provisioning uživatelů.

Stejně jako u dřívější metody je pro správnou funkčnost modulu nutné adekvátně nastavit projektovou bezpečnost. OIDC SSO pracuje se 3 modulovými rolemi: Administrator, User a Anonymous.

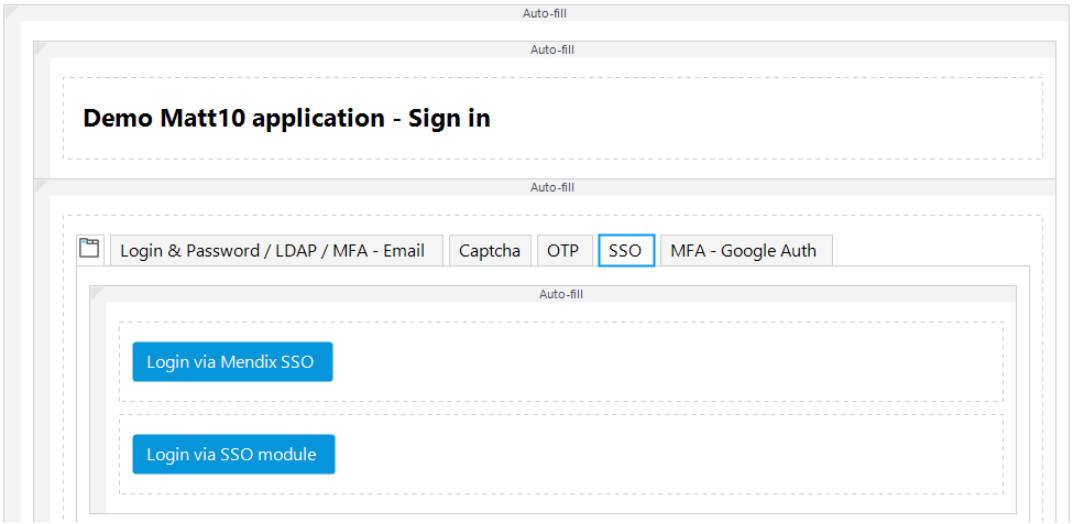
Implementace

Implementace nebyla tak jednoduchá, jako u předchozího modulu, avšak nebyla ani příliš složitá. Před stažením samotného modulu bylo nutné stáhnout celkem 5 jiných Mendix modulů kvůli závislostem:

- Encryption (v.8.0.3), (Mendix 2022l)
- Community Commons (v.9.0.2), (Mendix 2022k)
- Nanoflow Commons (v.2.6.1), (Mendix 2022m)
- Native Mobile Resources (v.3.13.1), (Mendix 2022n)
- Mx Model reflection (v.6.2.0), (Mendix 2020)

Moduly MxModel reflection a Encryption musely být ještě samostatně dokonfigurovány. Následně stačilo umístit tlačítko na login page, aby bylo umožněno samotné přihlašování[73], viz spodní tlačítko na Obrázek 30. Tlačítko „Login via Mendix SSO“ souvisí s předchozím modulem.

[72] Myšleno, že mu je vytvořen uživatelský účet.
[73] Modul defaultně pracuje s list view. Jelikož je zde využit pouze jeden identity provider, je zde tato funkcionalita redukována do tlačítka.



Obrázek 30: SSO Metody na login stránce (zdroj: Autor)

Posledním krokem byla úprava provisioning MF. Zde bylo nutné změnit používanou uživatelskou entitu na entitu využívanou v aplikaci. Byly proto vytvořeny kopie 2 MF, UserProvisioning_Custom a UserProvisioning_Implementation, které byly změněny a nahrazeny.

Tímto byly dokončeny úpravy kódu a následně bylo nutné aplikaci spustit a nakonfigurovat. Aplikace byla napojena na Google SSO OIDC IdP, kdy pro konfiguraci byl použit získaný Client ID, Client Secret a well-known URI. Konfigurace dostupná níže, viz Tabulka 9.

Tabulka 9: Konfigurace OIDC SSO (zdroj: Autor)

Název	**Hodnota**
Client ID	651858853347-rc87a16hgpufec0v43noaof2hj11e6dg.apps.googleusercontent.com
Client Secret	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
Authorization Endpoint	https://accounts.google.com/o/oauth2/v2/auth
Token Endpoint	https://oauth2.googleapis.com/token
User info Endpoint	https://openidconnect.googleapis.com/v1/userinfo
JWKS uri	https://www.googleapis.com/oauth2/v3/certs
Issuer	https://accounts.google.com
Scopes	Openid; email; profile

Pole, neuvedená v tabulce, byla zanechána prázdná.

Doba Relace

Podobné, jako u Mendix SSO. Je zde pouze vyšší volnost nastavení tokenů či jiné konfigurace dle možností IdP. Na rozdíl od Mendix SSO je zde možné provést odhlášení od IdP pomocí standardního Logout SSO endpointu, díky čemuž bude nutné při dalším uživatelově přihlášení opětovně zadat jméno a heslo. Dle možností IdP je zde možné nastavit dobu relací lépe a detailněji, jak v předchozím Mendix SSO.

Limitace a omezení

Pro tento modul jsou zásadní následující limitace a omezení (Mendix 2023g):

- Není možné využít pro nativní mobilní aplikace
- Není možné využít pro PWA – Progressive Web Apps
- Nejsou podporovány autentizační metody jako asymetrické klíče

5.1.6 LDAP

Poskytovatel a správa uživatelů

Pro hosting adresářového serveru byl použita služba „LDAP as a service“ od jumpcloud, (jumpcloud 2023a). Jedná se o bezplatný hosting LDAP serveru, kde je možné snadno nastavit server a spravovat uživatelské účty. Důvodem zvolení tohoto poskytovatele byla potřeba online hostovaného serveru, na který se bude možné připojit jak z lokální, tak z cloudové aplikace.

Konfigurace LDAP serveru nebyla obtížná, jelikož poskytovatel nabízí mnoho tutoriálů a dokumentace. Zároveň je již mnoho věcí přednastaveno. Byla potřeba jen nastavit tzv „binding“ uživatele, dále testovacího uživatele, skupinu pro tyto uživatele a následně je všechny propojit s „JumpCloud LDAP“, tedy samotným cloudovým LDAP serverem. Všechny potřebné odkazy a hodnoty pro napojení byly poté snadno získatelné buď z detailu cloudové služby či z obecné dokumentace.

Komunikace

Komunikace probíhá pomocí protokolu LDAP, kde modul podporuje i využívání LDAPS, tedy protokolu LDAP používajícím TLS pro šifrování komunikace a vyšší úroveň bezpečnosti. Volbu protokolu musí být provedena v rámci nastavení adresy serveru v konfiguraci.

Způsoby, jakými modul komunikuje se serverem jsou častokrát velmi excesivní. Ku příkladu bylo zaznamenáno, že pro autentizaci jednoho uživatele musí modul vykonat 7 separátních volání serveru. Na obrázku níže, viz Obrázek 31, je možné vidět celkově 5 bind akcí pro bind uživatele (System), 1 search akce, a 1 bind akce samotného uživatele, který se snaží přihlásit do aplikace (LDAPTest).

	Timestamp	Event Type	Result	Initiated By	Success
⌄	Feb 03, 2023 @ 10:13:12.98	ldap_bind	LDAP bind successful	LDAPTest	true
⌄	Feb 03, 2023 @ 10:13:12.89	ldap_bind	LDAP bind successful	System	true
⌄	Feb 03, 2023 @ 10:13:12.78	ldap_bind	LDAP bind successful	System	true
⌄	Feb 03, 2023 @ 10:13:12.64	ldap_srch	—	System	—
⌄	Feb 03, 2023 @ 10:13:12.59	ldap_bind	LDAP bind successful	System	true
⌄	Feb 03, 2023 @ 10:13:12.49	ldap_bind	LDAP bind successful	System	true
⌄	Feb 03, 2023 @ 10:13:12.34	ldap_bind	LDAP bind successful	System	true

Obrázek 31: LDAP komunikace (zdroj: Autor)

Bezpečnost

Modul efektivně funguje jako rozšíření základní Mendix Java login akce, která již byla představena v kapitole 5.1.1. Modul je přímo tak implementován. Zde se místo úpravy chování kontroly hesla přímo upraví celá metoda tak, aby autentizace probíhala přes LDAP.

Ve vztahu k bezpečnosti, tím je zaručeno, že metoda se chová velmi podobně jako přihlašování jménem a heslem. Sessions jsou handlovány stejně[74], hesla se ukládají stejným způsobem apod. Jediným rozdílem je, že zde chybí bezpečnostní mechanismy pro zablokování uživatelů, a tyto opatření se tedy musí buď dodatečně doplnit na straně klienta, nebo zařídit na straně serveru.

Modul podporuje 3 typy chování. Může pouze autentizovat uživatele na serveru, autentizovat uživatele na serveru a vytvořit uživatelský účet pokud ještě neexistuje, nebo pouze importovat uživatele a autentizaci neprovádět. V aplikaci je využit druhý typ, ale je vhodné, že je možné využít i jiné typy pro jiné scénáře.

Při tvorbě uživatelských účtů se jako Name použije uid získané z LDAP serveru a jako heslo je vygenerován 32 znakový řetězec s 5 číslicemi, 5 velkými písmeny a 5 znaky.

Implementace

Implementace této metody byla značně problematická. Hlavními důvody obtížnosti bylo, že metoda není již od verze Mx8 aktualizovaná či udržovaná a zároveň chybí jakákoliv detailnější dokumentace.

Aby se modul mohl použít v demo aplikaci, musel být prvně stažen ve starší verzi Mendixu[75]. Zde k němu byly zároveň staženy moduly Encryption a Community Commons, jelikož je na nich modul dependentní. Dále bylo nutné upravit Java kód několika metod, protože modul používá pro Mx9 již nepoužívané metody a nová verze modulu Encryption také pracuje

[74] Na straně serveru není nutné relace ukončovat či jinak řídit, jelikož tam ani nevznikají.
[75] Byla použita verze 8.18.10, ale bylo by možné použít jakoukoliv verzi Mx8.

s jinými metodami a třídami. Bylo nutné upravit celkem 4 Java soubory. K nalezení a opravení chyb velmi pomohlo Mendix Fórum, kde jeden z uživatelů, (De Gelder 2021), popsal jak část souborů upravit.

Takto připravený projekt byl následně konvertován do verze 9.18.0. kde bylo nutné přehodit page templates u obsažených stránek[76], přidat modulový „after start up“ MF do projektového „after start up“ MF, napojit konfigurační stránku do navigace a přidat administrátorskou modulovou roli projektové roli administrátora.

LDAP Modul funguje podobně jako autentizace pomocí vlastní Java akce, kdy mění fungování samotné login akce. Proto nebylo nutné výrazně měnit či připravovat login page. Mendix zároveň zakazuje používání vícero autentizačních widgetů na jedné stránce, takže byla tato metoda spojena s metodou přihlašování heslem.

Tím byla pokryta úprava kódu a dále bylo nutné aplikaci spustit a nakonfigurovat. Nejdůležitější nastavené hodnoty jsou zaneseny v Tabulka 10. Správně nakonfigurovat aplikaci bylo složité. Není zde dostatek dokumentace a modul se v dosti situacích chová specificky, tj. odlišně od standardních LDAP řešení. Výsledné hodnoty byly jediné, které se dají nastavit aby modul fungoval korektně.

Tabulka 10: Konfigurace LDAP (zdroj: Autor)

Název	**Hodnota**
Server address	ldaps://ldap.jumpcloud.com:636
LDAP root directory	ou=Users, o=63d7810bca81c045134081ea, dc=jumpcloud, dc=com
Ldap System username	uid=System, ou=Users, o=63d7810bca81c045134081ea, dc=jumpcloud, dc=com
LDAP type	Authenticate users against the AD and create the user account if it doesn't exist
Map users to	Demo.CustomAccount
Synchronization method	Synchronize by path
Ldap Login field name	uid
UserPath	ou=Users, o=63d7810bca81c045134081ea, dc=jumpcloud, dc=com
Credentials validation frequency	On every login, but store information during session

[76] Důvodem je, že mezi verzemi Mx8 a Mx9 došlo k výrazným UI změnám.

Limitace a omezení

Hlavní limitací této metody je fakt, že není možné používat jiné autentizační metody společně s touto metodou. Dojde zde k takové úpravě login mechanismu, že pokud nebude uživatel dostupný na LDAP serveru, nebude se moci nijak přihlásit. Jedinou výjimkou jsou uživatelé s rolemi, které jsou specifikovány jako role nepoužívající LDAP. Tito uživatelé se mohou přihlásit klasicky pomocí jména a hesla.

Doba Relace

Stejná, jako v případě přihlašování jménem a heslem. Uživatelům se netvoří relace na straně serveru.

5.1.7 Více faktorová autentizace

V následujících 2 podkapitolách 5.1.8 a 5.1.9 jsou rozebrány další 2 způsoby autentizace pomocí OTP v kontextu více faktorové autentizace.

5.1.8 Google Authenticator

Vybraný Mendix modul implementuje Java knihovnu warrenstrange.GoogleAuthenticator (Strange 2023), díky které je možné jednoduše implementovat TOTP dle RFC 6238 (M'Raihi et al. 2011). Jedná se o modul silně zaměřený pouze na propojení a validování TOTP, neobsahuje tedy žádné entity či MF ke konfiguraci, bezpečnostní opatření či jiné doplňky[77].

Knihovna generuje šestimístné TOTP kódy, platné po dobu 30 sekund a umožňuje propojení s mobilní aplikací Google Authenticator pomocí odkazu – QR Kódu.

Uložení dat

V rámci aplikace je nutné ukládat pouze Secret key uživatelů. Tyto klíče se ukládají v prostém textu a nedochází k žádnému šifrování. V případě nutnosti by bylo možné jej doplnit.

[77] Např. vestavěné logování či možnost vyžadování ověření TOTP jen jednou v rámci určitého časového úseku apod.

Komunikace

Pro správné fungování modulu stačí komunikace runtime a klienta. Není potřeba komunikace s žádnými třetími stranami. K přenosu dat dochází vždy při vygenerování odkazu pro propojení mobilní aplikace a pro validaci generovaných TOTP.

Bezpečnost

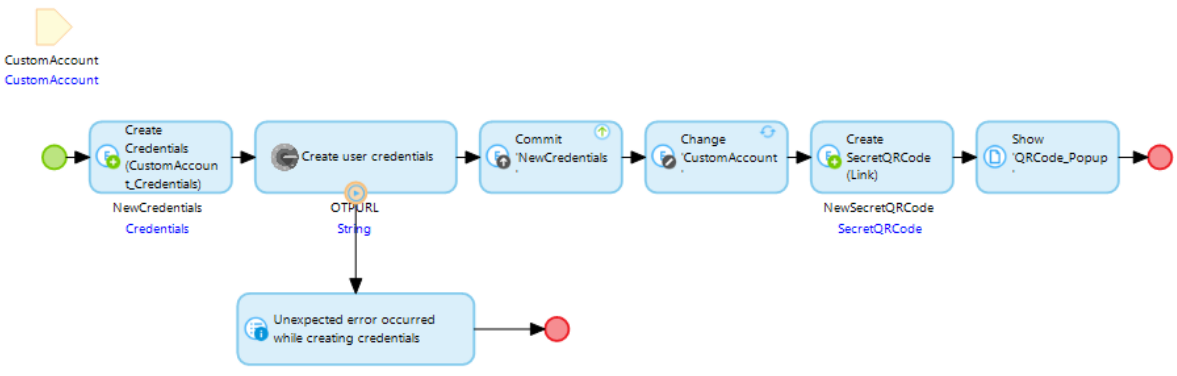
Modul je bezpečnostně na projekt nenáročný. Nezamezuje používání jiných autentizačních modulů a metod a zároveň nedefinuje žádné nové modulové role. Implementace i používání je bezproblémová. Jediným bezpečnostním rizikem je používání secret key, který, pokud by byl kompromitován, by mohl umožnit útočníkovi se jednoduše vydávat za uživatele.

Implementace

Modul poskytuje 2 Java akce, jednu na vytvoření Secret key a odkazu pro propojení s mobilní aplikací Google Authenticator, druhou pro validaci kódu pocházející z aplikace. Sám o sobě zde není definován žádný doporučený způsob implementace a ani není nutné měnit AfterStartUp MF.

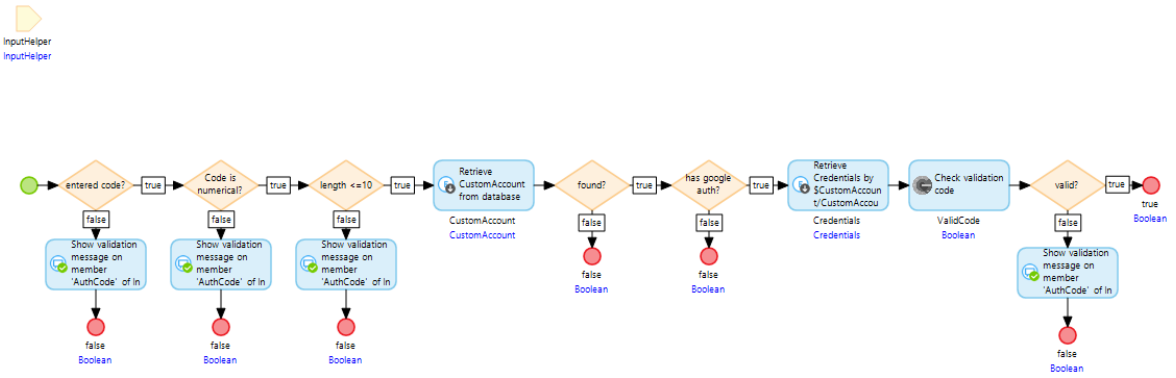
Modul je tedy možné využít v aplikaci jakkoliv. V Demo aplikaci byl implementován jako druhý faktor při přihlašování jménem a heslem s tím, že aby uživatel mohl tento druhý faktor používat, musí se prvně standardně přihlásit a spustit si jej v rámci nastavení vlastního profilu.

Spouštění probíhá způsobem, že uživatel stiskne tlačítko „Setup Google Auth“, které spustí MF, obsahující první Java akci modulu, viz Obrázek 32. V tomto MF se prvně vytvoří entita Credentials, která se asociuje s entitou uživatele a do které Java akce uloží Secret key do atributu „ga_SecretKey“. Tyto Credentials se následně uloží do databáze a uživateli se otevře stránka, kde za použití specifického widgetu je zobrazen QR kód, který si uživatel může naskenovat do mobilní aplikace. Jindy než v tomto případě se uživatel ke QR kódu, a obecně Secret key nedostane.



Obrázek 32: Registrace zařízení (zdroj: Autor)

Samotné přihlášení je následně implementováno tak, že uživatel je po zadání validního jména a hesla vyzván k zadání TOTP z aplikace. Pokud uživatel vloží validní kód, což je validováno za pomocí druhé Java akce modulu, bude přihlášen. Viz Obrázek 33.



Obrázek 33: Validace TOTP (zdroj: Autor)

Pokud by uživatel nechtěl dále používat tento způsob autentizace, nebo pokud by chtěl používat jiné zařízení, stačí pouze smazat jeho asociované Credentials opět pomocí tlačítka z nastavení profilu.

5.1.9 Email

Použitý modul poskytuje vícero rozhraní a možností pro implementaci více faktorové autentizace pomocí OTP. Modul umožňuje umístit do login mechanismu MF na určení, jestli se má více faktorová autentizace použít, dále MF na generování a odeslání OTP, a následně MF pro validaci přijatého OTP.

Jakou logiku použít a jaký způsob generování či ověřování je volitelné a lze zde použít mnoho variant. Sám modul poskytuje tři způsoby jako příklady. Prvním je SMS pomocí služby Twilio, druhým je email pomocí API služby Sendgrid a třetím je již dříve použitá aplikace Google Authenticator.

Jelikož se jedná o úpravu výchozího login mechanismu, je nutné počítat, že tento modul není možné kombinovat s jinými moduly, které jej taky modifikují. Na druhou stranu je zde poskytována řada dalších funkcionalit, které pro moduly, jako např. modul pro Google Authenticator z minulé kapitoly, nejsou snadno proveditelné. Modul umožňuje nakonfigurovat:

- Spuštění modulu
  - Možné určit pomocí boolean konstanty, zdali bude modul spuštěný či vypnutý
- Max počet pokusů MFA
  - Maximální počet pokusů o přihlášení za použití MFA, než dojde k zablokování
- Max počet pokusů o přihlášení
  - Maximální počet pokusů o přihlášení, než dojde k zablokování
- Čas odblokování MFA
  - Čas v minutách, po jehož uplynutí bude blokovaný uživatel odblokován, jestliže byl zablokován kvůli překročení Max počtu pokusů MFA
- Čas odblokování

- Čas v minutách, po jehož uplynutí bude blokovaný uživatel odblokován, jestliže byl zablokován kvůli překročení Max počtu pokusů o přihlášení

Uložení dat

Modul ukládá odeslané kódy párované s uživatelskými jmény v rámci vlastní entity MFAmodule.MFA. Data nejsou nijak šifrována. Tato entita je asociována s relací anonymního uživatele předtím, než se přihlásí, přičemž pokud dojde k přihlášení, nebo opuštění stránky a následným zrušením této relace, entita MFA se smaže z databáze.

Uživateli se data z této entity nijak nezobrazují. Pouze dostane OTP na své zařízení či emailem.

Komunikace

Kromě komunikace klienta s Mendix runtime za pomoci HTTPS zde dochází ke komunikaci se externími stranami při odesílání OTP uživateli. V rámci příkladů, co poskytuje modul, by se mohlo jednat např. o standardní REST API komunikaci, kde pomocí metody HTTP POST jsou odesílána data o příjemci, odesílateli a obsahu zprávy poskytovateli emailové, či SMS služby. Autentizace u těchto poskytovatelů probíhá buď pomocí bearer tokenu či pomocí soukromého API klíče.

Obecně ale záleží na implementaci a výběru poskytovatele. Modul pouze poskytuje příklady a poskytuje volnost implementovat komunikaci s jakýmkoliv generátorem OTP a jakoukoliv službou k jejich odeslání, validování či jinému zpracování.

Bezpečnost

Modul obsahuje 3 modulové role: Administrator, User, Anonymous. Ty je nutné propojit s rolemi projektu, primárně roli anonyma, pod kterou spadá všechna autentizační logika. Pro Administrátora je poskytován omezený přístup k informacím o přihlašování a pro User je dostupná pouze logika odhlášení.

Jako jediný analyzovaný modul jsou v tomto poskytovány Unit testy, pomocí kterých je možné testovat správné chování nastavených metod. Je možné testovat přihlášení, validaci, blokování, odblokování, odhlášení a správnost konfigurace modulu.

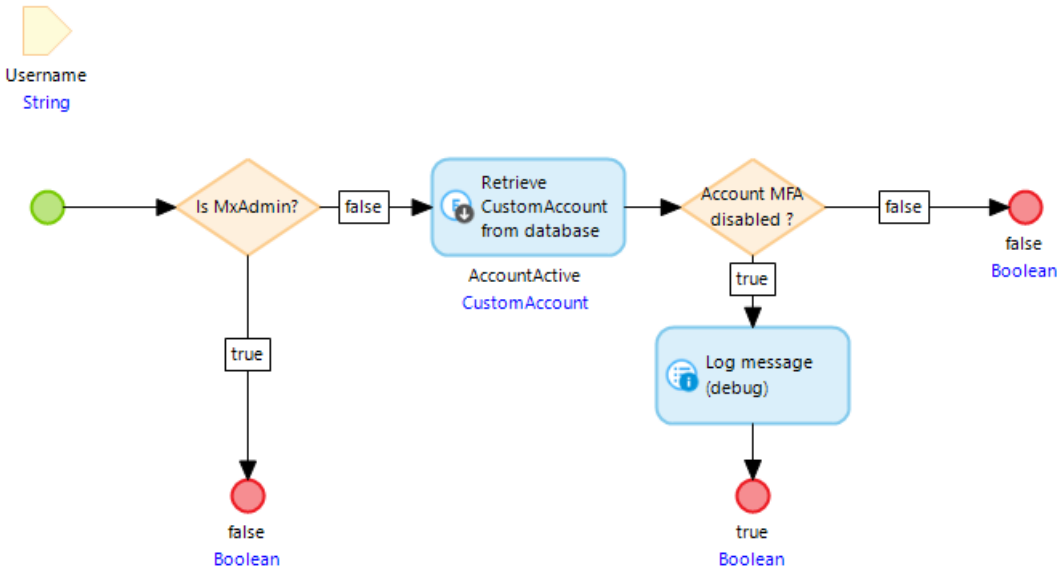
Doba Relace

Řídí se stejnými pravidly jako v případě metody jména a hesla.

Implementace

Celá implementace modulu spočívala v nastavení jeho spouštěcí části do AfterStartUp MF a následně nastavením tří, dříve zmíněných, MF pro určení použití MFA, generování kódu a pro jeho validaci.

Pro určení použití MFA byl použit modifikovaný vzorový MF. Byl upraven aby pracoval s uživatelskou entitou aplikace a aby MFA používali jen ti uživatelé, kteří mají specificky zapnuté MFA, což bylo určeno pomocí nového boolean atributu. Tento MF, viz Obrázek 34, vrací „True“, pokud uživatel nemá či nemusí MFA používat, a „False“, pokud se má MFA použít, tedy dojde ke spuštění dodatečné logiky.



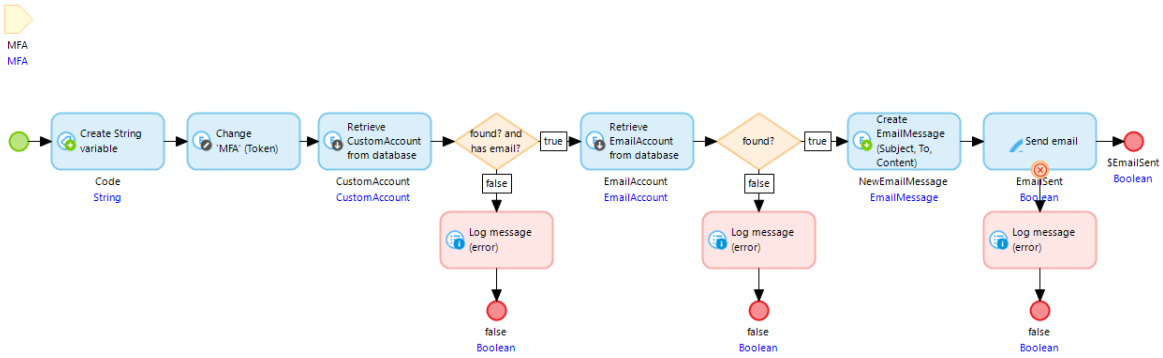
Obrázek 34: MF určující použití MFA (zdroj: Autor)

Pro MF a generování OTP byl použit MF, viz Obrázek 35, který byl také vytvořen modifikací vzoru. Jedná se o jednoduchý proces tvorby kódu, který se následně odešle emailem uživateli[78]. Kód je generován za pomoci výrazu:

```
toString(round(random() * 150000)) + 100000
```

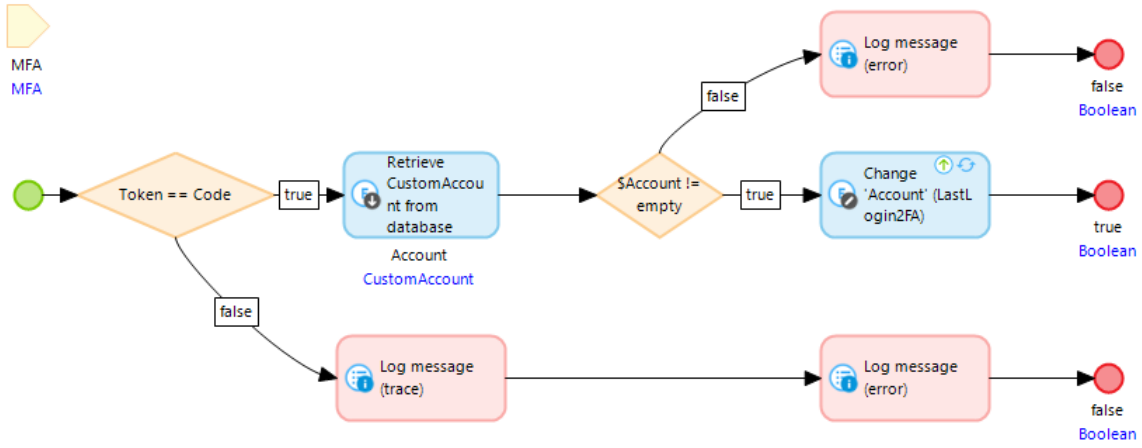
kde random() generuje náhodný decimal v rozmezí 0 až 1 s přesností na 17 desetinných míst a round() zaokrouhluje na nejbližší celé číslo. Je zvláštní, že konečných 100000 se zde přičítá jako string a ne jako integer, což je možná chyba ze strany autorů modulu. Možné výsledné hodnoty tedy jsou 0 až 150000 s dodatkem 100000 na konci. Výsledný kód se poté odesílá uživateli.

[78] K odesílání emailů musel být stažen a nakonfigurován emailový modul Email Connector, (Mendix 2023b).



Obrázek 35: MF pro generování a odeslání OTP (zdroj: Autor)

Posledním je MF pro validaci příchozího kódu. Vrací „True“, pokud validace proběhla v pořádku, a „False“, pokud ne. Viz Obrázek 36.



Obrázek 36: Validační MF (zdroj: Autor)

Posledním krokem implementace modulu byla příprava uživatelských stránek, kde si můžou jednotliví uživatelé zapnout či vypnout MFA pro svůj účet. Dále by bylo možné konfigurovat modul pomocí konstant, zmíněných na začátku této kapitoly, ale to zde nebylo nutné.

Limitace a omezení

Ačkoliv tento modul funguje primárně pomocí změn v defaultní login akci, není možné používat společně s defaultní login stránkou. Dle poskytovatelů se jedná o komplexní a obtížně zabezpečitelný problém a je proto nutné používat na stránce vlastní.

5.2 Nativní metody autentizace

5.2.1 Biometrie

Metody biometrické autentizace, dostupné v modulech na Mendix Marketplace, jsou specificky navržené jen pro ověření, že se zařízením pracuje jeho majitel (Mendix 2022n).

Aktuálně analyzovaný modul[79], NativeMobileResources poskytuje možnost využití 2 JS akcí, první pro určení, zdali je aktuální zařízení schopné biometricky autentizovat uživatele, druhá pro vyvolání a vyhodnocení samotné autentizace.

Problémem pro reálné využití tedy je, že akce podporují pouze scénář, kdy je nutné nějak ověřit, že s mobilem pracuje jeho majitel, či osoba, která v něm má uložené svůj otisk prstu/ FaceID či jinou podobu biometrického otisku. Metody vrací pouze True/False hodnoty, pokud autentizace prošla avšak neposkytují žádné řešení k tomu, jak např. následně uživatele přihlásit. Pokud by vývojář tedy chtěl implementovat přihlášení jen pomocí otisku prstu, musel by si někde uložit jméno a heslo uživatele, které by následně použil pro přihlášení v případě úspěšné biometrické autentizace.

Uložení dat

JS metody samy žádná data neukládají. Není tedy možné se přímo u nich zabývat problematikou bezpečnosti uložení dat. Na druhou stranou je v prostředí Mendixu nutné pro přihlášení použít v rámci NF jméno a heslo. Je tedy nutné tento problém vyřešit.

Jelikož se zde pracuje s JS akcemi, které lze použít pouze v NF, není možné vytvořit podobné řešení jako v případě OTP, kde by stačilo vhodně upravit samotnou login akci. A i kdyby to možné bylo, bylo by nutné nějak propojit údaje o zařízení či biometrické údaje uživatele s jeho profilem, což ovšem v rámci těchto metod je obtížné.

Aby bylo možné zajistit bezpečné, či žádné ukládání dat, jsou zde možná dvě řešení. Buď metodu použít jako druhý faktor přihlášení, kdy uživatel bude muset jméno a heslo vyplnit pokaždé a biometrie se použije jen jako dodatečné ověření, že se z mobilu přihlašuje opravdu osoba, která by k němu měla mít přístup[80]. V tomto případě by samotné přihlášení proběhlo pomocí vyplněných údajů. Druhým řešením je uložit přihlašovací údaje na zařízení a použít je pro přihlášení, pokud se uživatel autentizuje pomocí biometrie. Efektivně by šlo o jakýsi biometrický zámek pro přístup k reálným údajům pro přihlášení. Zde je ale otázka bezpečnosti uložení citlivých dat na zařízení uživatele, která ačkoliv mohou být šifrována, představují nové bezpečnostní zranitelnosti.

Bezpečnost

JS akce implementují knihovnu React Native Touch ID (Kadhom 2018), která již není podporována a udržována svými autory a její poslední verze pochází až z roku 2018. To je problematické, jelikož v případě nalezení nových zranitelností, nedojde k aktualizaci knihovny. Aktuálně knihovna neobsahuje žádné známé zranitelnosti (Snyk 2023).

---

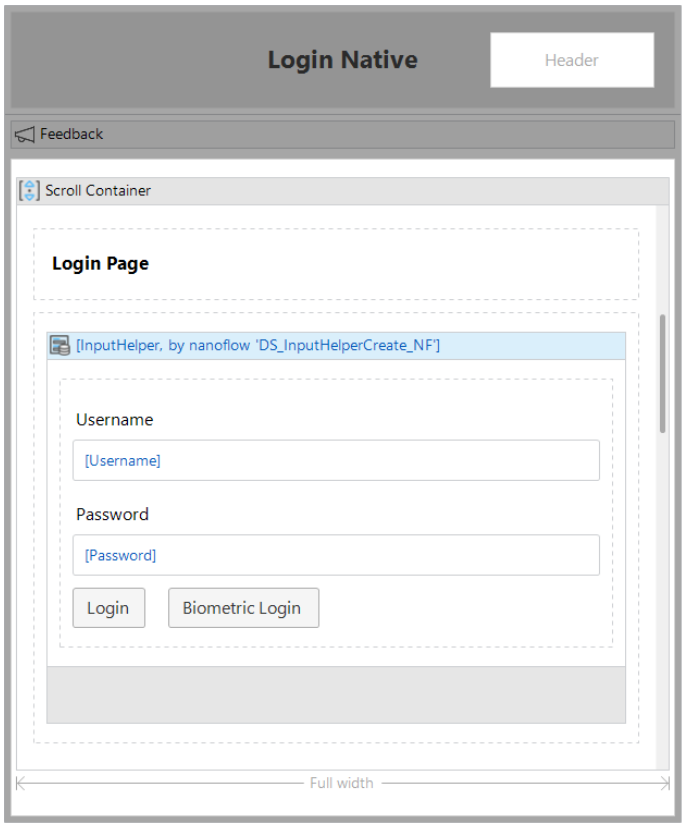
[79] Dříve existoval ještě modul pro hybridní aplikace. Ten ale již vyšel z podpory, jako celé hybridní aplikace v Mendixu.
[80] Je možné si uložit nějaké informace o zařízení k uživatelově účtu a ty následně kontrolovat, např. tím že by bylo povoleno přihlašování jen z těchto zařízení apod.

Knihovna je také kritizována kromě omezené podpory, nutnosti uložení dat a způsobu autentizace[81] i pro svou implementaci (Dinh 2019) (Avertra 2022). Mnoho autorů avizuje, že pro vyšší bezpečnost by bylo vhodné používat knihovny implementující key chain, jako např. v tomto případě „react-native-keychain“

Implementace

Prvním krokem implementace bylo vytvoření nativního profilu v rámci navigace projektu a vytvoření 2 nových nativních stránek – domácí stránku a login stránku, viz Obrázek 37. Nativní aplikace musí používat stránky postavené na nativních šablonách a není tedy možné používat stránky z klasického, responsivního, profilu.

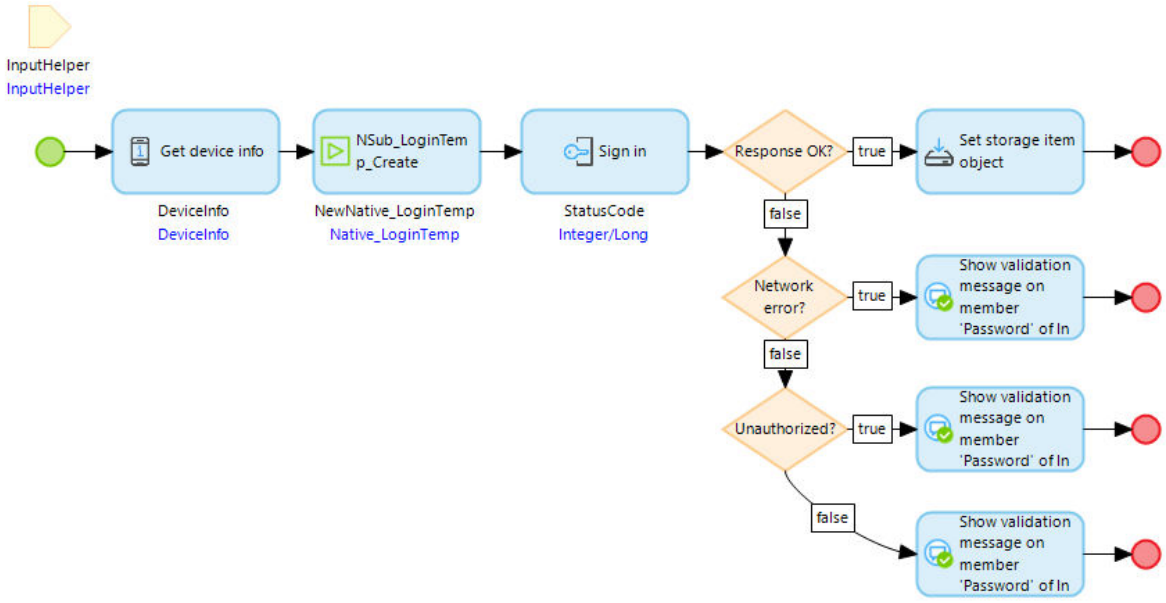
Implementace byla provedena podobně, jako ve článku (Tran 2021). Tento přístup byl zvolen, jelikož není dostupná téměř žádná dokumentace o dříve zmíněných JS akcích a zároveň článek popisuje jediné efektivní řešení, které jde s danými možnostmi implementovat. Nevýhodou řešení, a možným bezpečnostním problémem je, že je nutné si na mobilním zařízení uložit přihlašovací údaje, bez kterých by nebylo možné se přihlásit pouze za použití biometrie.



Obrázek 37: Nativní login stránka (zdroj: Autor)

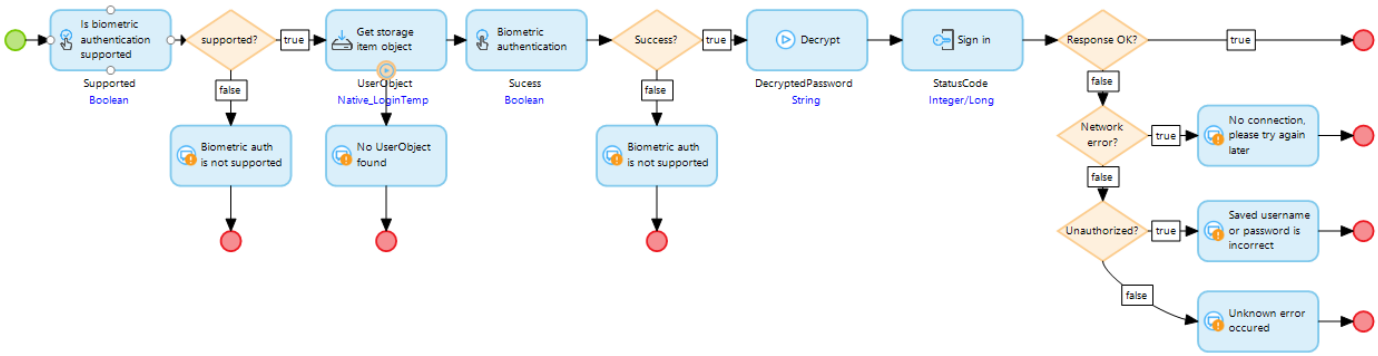
[81] Metody knihovny vrací pouze true/false, oproti např. kryptografickému klíči či jinému objektu (Lardinois 2021).

Login stránka obsahuje 2 tlačítka, jedno pro spuštění klasického přihlášení pomocí jména a hesla a druhé pro biometrické přihlášení. Idea řešení je, že při spuštění prvního NF, viz Obrázek 38, se při úspěšném přihlášení uloží jméno, heslo a informace o zařízení do local storage jako „Native_LoginTemp“ entita. Tato entita se dále využije v následujících, biometrických přihlášeních. Pro vyšší bezpečnost je uživatelské heslo šifrováno.



Obrázek 38: Nativní login pomocí jména a hesla (zdroj: Autor)

V rámci druhého NF dojde ke kontrole, zdali je možné se přihlásit pomocí biometrie, což spočívá v možnostech mobilního zařízení. Zároveň je nutné, aby v local storage zařízení již byly uloženy přihlašovací údaje. Pokud jsou kontroly splněny, dojde k vyzvání na biometrickou autentizaci, která, pokud uspěje, načte přihlašovací údaje z local storage a pokusí se pomoci nich uživatele přihlásit.



Obrázek 39: Nativní login pomocí biometrie (zdroj: Autor)

6 Porovnání metod

6.1 Dle použitelnosti

6.1.1 Výsledky testování

Tabulka 11: Výsledky testování použitelnosti (zdroj: Autor)

Metoda	Použitelné na více zařízeních	Počet akcí	Doba k aut.	Potřeba účtu v jiné aplikaci	Uživatelský komfort	Spoleh.	Časové vypršení relace a reauten.
Jméno a heslo	Ano	2	6,71 s	Ne	Vysoký	Vysoká	Globální
Jméno a heslo & Captcha	Ano	3	11,50 s	Ne	Střední	Vysoká	Globální
OTP	Ano	5	14,28 s	Ne	Střední	Vysoká	Globální
Mendix SSO	Částečně	3	13,00 s	Ano	Vysoký	Vysoká	Pevně stanovené
OIDC SSO	Částečně	3	12,31 s	Ano	Vysoký	Vysoká	Individuální
LDAP	Ano	2	8,22 s	Ano	Vysoký	Vysoká	Globální
MFA – Google auth	Ano	5	12,75 s	Ne	Střední	Vysoká	Globální
MFA – Email	Ano	5	16,78 s	Ne	Střední	Vysoká	Globální
Biometrie	Ne	2	5,22 s	Ne	Vysoký	Vysoká	Globální

Dodatek – počet akcí

Jméno a heslo

1. Vyplnění přihlašovacích údajů
2. Stisknutí sign-in tlačítka

Jméno a heslo & Captcha

1. Vyplnění přihlašovacích údajů
2. Vyplnění Captcha okna
3. Stisknutí sign-in tlačítka

Pozn. – V tomto případě byly testovány případy, kdy se uživateli zobrazí Captcha okno, které musí vyplnit. Toto okno se otevře až po několika pokusech o přihlášení a nedávalo by zde smysl testovat Captcha metodu bez něj, jelikož poté by výsledky byly téměř identické s login metodou bez ní.

OTP

1. Vyplnění přihlašovacích údajů
2. Stisknutí sign-in tlačítka
3. Otevření mobilu/emailu/místa, kam uživateli přijde OTP kód
4. Přepsání kódu
5. Stisknutí sign-in tlačítka

Mendix SSO

1. Stisknutí sign-in via SSO tlačítka
2. Přihlášení u IdP
3. Odkliknutí Autorizování aplikace

OIDC SSO

1. Stisknutí sign-in via SSO tlačítka
2. Přihlášení u IdP
3. Odkliknutí Autorizování aplikace

LDAP

1. Vyplnění přihlašovacích údajů
2. Stisknutí sign-in tlačítka

MFA – Google auth

1. Vyplnění přihlašovacích údajů
2. Stisknutí sign-in tlačítka
3. Otevření mobilní aplikace
4. Přepsání kódu
5. Stisknutí sign-in tlačítka

MFA – Email

1. Vyplnění přihlašovacích údajů
2. Stisknutí sign-in tlačítka
3. Otevření emailu
4. Přepsání či překopírování kódu

5. Stisknutí sign-in tlačítka

Biometrie

1. Stisknutí tlačítka pro použití biometrie
2. Provedení biometrického přihlášení pomocí prstu či face ID

Pozn. – při testování biometrie se pracuje s případy, kdy již uživatel měl v local storage zařízení uloženy přihlašovací údaje. Pokud by tomu tak nebylo, nešlo by testovat biometrické přihlašování ale pouze přihlašování jménem a heslem v nativní aplikaci.

Dodatek – doba k autentizaci

Každá metoda byla testována desetkrát a finální hodnota je průměrem všech testování. Pro testování přihlašování s lokálními uživateli byl použit účet Test s heslem „Password123,“. Pro LDAP účet LDAPTest se stejným heslem a pro zbytek mé osobní Google či Mendix účty. V Tabulka 12 jsou znázorněny celkové výsledky.

Tabulka 12: Doba autentizace jednotlivých metod v sekundách (zdroj: Autor)

Metoda	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Průměr**
Jméno a heslo	7,6	7,2	6,2	6,4	5,9	6,7	6,5	7,2	6,5	6,9	**6,71**
Jméno a heslo & Captcha	10,1	10,2	14,2	11,1	12,1	10,5	13	11,3	12,2	10,3	**11,50**
OTP	17,0	14,5	14,8	10,5	13	18,6	10,9	13,8	13,6	16,1	**14,28**
Mendix SSO	14,3	12,3	12,2	12,5	13,4	12,9	14,2	12,6	12,6	13,0	**13,00**
OIDC SSO	12,6	12,1	11,5	10,9	13,2	13,2	12,4	10,8	13,4	13,0	**12,31**
LDAP	8,1	8,5	8,8	9,8	7,3	8,7	8,2	7,4	7,3	8,1	**8,22**
MFA – Google auth	11,9	14,5	12,4	12,7	11,9	12,2	12,3	13,6	14,1	11,9	**12,75**
MFA – Email	17,2	27,0	15	13,5	17,0	24,0	15,1	11,2	13,0	14,8	**16,78**
Biometrie	5,6	5,3	4,9	5,1	4,8	5,8	4,9	4,9	5,6	5,3	**5,22**

6.1.2 Výsledná kritéria

Tabulka 13: Výsledná kritéria použitelnosti (zdroj: Autor)

Metoda	Použitelné na více zařízeních	Počet akcí	Doba k aut.	Potřeba účtu v jiné aplikaci	Uživatelský komfort	Spoleh.	Časové vypršení relace a reauten.
Jméno a heslo	Ano	Pár	Krátká	Ne	Vysoký	Vysoká	Globální
Jméno a heslo & Captcha	Ano	Několik	Střední	Ne	Střední	Vysoká	Globální
OTP	Ano	Více	Spíše delší	Ne	Střední	Vysoká	Globální
Mendix SSO	Částečně	Několik	Střední	Ano	Vysoký	Vysoká	Pevně stanovené
OIDC SSO	Částečně	Několik	Střední	Ano	Vysoký	Vysoká	Individuální
LDAP	Ano	Pár	Střední	Ano	Vysoký	Vysoká	Globální
MFA – Google auth	Ano	Více	Střední	Ne	Střední	Vysoká	Globální
MFA – Email	Ano	Více	Spíše delší	Ne	Střední	Vysoká	Globální
Biometrie	Ne	Pár	Krátká	Ne	Vysoký	Vysoká	Globální

6.1.3 Výsledná úroveň

Tabulka 14: Výsledná úroveň dle použitelnosti (zdroj: Autor)

Metoda	Úroveň použitelnosti
Jméno a heslo	4
Jméno a heslo & Captcha	3
OTP	2
Mendix SSO	3

OIDC SSO	4
LDAP	4
MFA – Google auth	2
MFA – Email	2
Biometrie	3

6.2 Dle bezpečnosti

6.2.1 Výsledná kritéria

Tabulka 15: Výsledná kritéria bezpečnosti (zdroj: Autor)

Metoda	**Bezpečnost a charakter dat při přenosu**	**Bezpečnost uložení dat**	**Odolnost vůči útokům**	**Rizika technologií a architektury**	**Stand.**
Jméno a heslo	Vysoká	Vysoká	Celková	Nízká	Vysoká
Jméno a heslo & Captcha	Vysoká	Vysoká	Celková	Nízká	Vysoká
OTP	Vysoká	Vysoká	Celková	Nízká	Vysoká
Mendix SSO	Vysoká	Vysoká	Celková	Nízká	Vysoká
OIDC SSO	Vysoká	Vysoká	Celková	Nízká	Vysoká
LDAP	Vysoká	Vysoká	Částečná	Střední	Vysoká
MFA – Google auth	Vysoká	Vysoká	Celková	Nízká	Vysoká
MFA – Email	Vysoká	Vysoká	Celková	Nízká	Vysoká
Biometrie	Vysoká	Střední	Celková	Střední	Vysoká

Dodatek – Bezpečnost uložení dat

Kritérium uložení dat bylo kontrolováno stažením celé aplikační databáze a průzkumem jednotlivých dat uložených v příslušných tabulkách. Všechna důležitá data se šifrují, až na výjimky ve 2 metodách:

- OIDC SSO nešifruje jiné, než Access tokeny
- Biometrie šifruje hesla, ale ukládá je na samotném zařízení

Dodatek – Odolnost vůči útokům

Všechny metody aplikují dostatečná opatření proti možným útokům, kromě metody LDAP, která je zranitelná vůči útokům hrubou silou.

Dodatek – Rizika technologií a architektury

Metoda LDAP není aktualizována či záplatována od verze MX8. Dále struktura a chování celé metody je dosti složitá a zbytečně komplexní, kdy často poté dochází k excesivním akcím, jako například 7 separátních volání serveru pro pouhé jedno přihlášení uživatele. Metoda by mohla být implementována bez zásahu do defaultní login akce, jak je implementován OIDC SSO modul. Tím by se značně snížila komplexita celého modulu.

Biometrie používá nebezpečné, neaktualizované knihovny, viz podkapitola 5.2.1.

Dodatek – Standardizace

Všechny metody jsou standardizovány vícero organizacemi. Mnoho z nich je standardizováno v RFC, NIST či ISO/IEC. Dále existuje mnoho best practices a studií o těchto metodách, například ze strany Auth0, FIDO, OWASP či jiných organizací z této oblasti.

6.2.2 Výsledná úroveň

Tabulka 16: Výsledná úroveň dle bezpečnosti (zdroj: Autor)

Metoda	Úroveň bezpečnosti
Jméno a heslo	5
Jméno a heslo & Captcha	5
OTP	5
Mendix SSO	5
OIDC SSO	5

LDAP	3
MFA – Google auth	5
MFA – Email	5
Biometrie	3

6.3 Dle nákladů a znalostí

6.3.1 Výsledná kritéria

Tabulka 17: Výsledná kritéria nákladů a znalostí (zdroj: Autor)

Metoda	**Dodatečné náklady provozu**	**Čas implementace a nutné lidské zdroje**	**Závazky vznikající z implementace**
Jméno a heslo	Žádné	Nízké	Nevznikají
Jméno a heslo & Captcha	Žádné	Nízké	Vznikají
OTP	Nízké	Nízké	Vznikají
Mendix SSO	Žádné	Nízké	Nevznikají
OIDC SSO	Nízké	Nízké	Vznikají
LDAP	Nízké	Střední	Vznikají
MFA – Google auth	Žádné	Nízké	Vznikají
MFA – Email	Nízké	Nízké	Vznikají
Biometrie	Žádné	Nízké	Nevznikají

Dodatek – Dodatečné náklady provozu

V případě metod OTP a MFA-Email je nutné mít zařízený emailový hosting pro rozesílání OTP kódů. Náklady na tento hosting ovšem nejsou tak vysoké, aby dosáhly na vyšší kategorii kritéria.

Metody LDAP a OIDC SSO potřebují hostované servery pro správu uživatelů. Ani v jednom případě se avšak nejedná o vysoké částky. V případě LDAP se cena obvykle pohybuje kolem 2 až 3 USD za uživatele či méně[82]. V případě OIDC SSO jsou ceny výrazně nižší, např. v případě poskytovatele Google by 100 uživatelů stálo provozovatele aplikace maximálně pár desítek USD (Google Cloud 2023a).

Ostatní metody nejsou finančně náročné nijak, jelikož jsou kompletně pokryty samotnou licencí Mendixu. Jedinou výjimkou je Captcha, která sice jako samotná služba placená být může, avšak aktuálně je dle podmínek Google reCaptcha používat službu až do 1 000 000 requestů měsíčně zdarma (Google Cloud 2023b).

Dodatek – Čas implementace a nutné lidské zdroje

Všechny metody kromě LDAP nebylo obtížné implementovat. Ačkoliv někde bylo zmatečné metodu správně nakonfigurovat či integrovat, nešlo o velké problémy a často je bylo možné vyřešit pomocí tutoriálů, vzorů či dokumentace. Zkušený vývojář by měl být schopen tyto metody implementovat do 0,5 MD.

V případě LDAP je velkým problémem neaktuálnost modulu, nedostačující dokumentace a komplexnost modulu. Při implementaci nejde kvůli těmto problémům postupovat tak rychle, jako u ostatních metod a zároveň je nutné postupovat opatrně a více času věnovat testování kvůli možným problémům při upgrade verzí. Pro urychlení a zkvalitnění implementace je vhodné, aby vývojář znal, jak upravit Java login akci v Mendixu a zároveň aby se orientoval jak ve struktuře projektů Mx8 i Mx9 a rozuměl jejich rozdílům a procesu upgrade na Mx9. I za takových podmínek může ovšem implementace metody zabrat kolem 1 MD.

Dodatek – Závazky vznikající z implementace

V případě následujících metod je nutné použít externí služby:

- Captcha
- OTP
- OIDC SSO
- LDAP
- MFA – Google Auth
- MFA – Email

Captcha potřebuje nastavení reCaptcha klíčů přímo ve zdrojovém kódu. Tyto kódy je nutné získat od externího poskytovatele, jako je v případě demo aplikace Google.

OIDC SSO a LDAP metody pracují s externími účty, proto je pochopitelné že je nutné zřízení externích projektů či úložišť u externích služeb.

---

[82] Např. v případě (jumpcloud 2023b) či (Foxpass 2023)

Obě MFA metody pracují s externími zařízeními či účty pro generaci či přijímání OTP kódů. V prvním případě je nutné využívat aplikaci Google Authenticator, ve druhém emailového klienta, či ve speciálních případech SMS službu či jiné služby v podobě API.

Ostatní metody pracují pouze s prvky, které jsou nalezitelné a využitelné přímo v platformě Mendix.

6.3.2 Výsledná úroveň

Tabulka 18: Výsledná úroveň dle nákladů a znalostí (zdroj: Autor)

Metoda	**Úroveň nákladů a znalostí**
Jméno a heslo	5
Jméno a heslo & Captcha	4
OTP	3
Mendix SSO	5
OIDC SSO	3
LDAP	3
MFA – Google auth	4
MFA – Email	3
Biometrie	5

7 Hodnocení a řazení

7.1 Tabulka

V Tabulka 19 níže jsou zaneseny celkové výsledky kritérií jednotlivých metod. V Tabulka 20 jsou hodnoty kritérií zprůměrovány a každé metodě je přiřazeno výsledné pořadí.

Tabulka 19: Výsledné hodnocení metod (zdroj: Autor)

Metoda	**Použitelnosti**	**Bezpečnost**	**Náklady a znalosti**
Jméno a heslo	4	5	5
Jméno a heslo & Captcha	3	5	4
OTP	2	5	3
Mendix SSO	3	5	5
OIDC SSO	4	5	3
LDAP	4	3	3
MFA – Google auth	2	5	4
MFA – Email	2	5	3
Biometrie	3	3	5

Tabulka 20: Pořadí metod (zdroj: Autor)

Metoda	**Průměr kritérií**	**Pořadí**
Jméno a heslo	4,67	1.
Jméno a heslo & Captcha	4,00	3. – 4.
OTP	3,34	7. – 9.
Mendix SSO	4,34	2.
OIDC SSO	4,00	3. – 4.

LDAP	3,34	7. – 9.
MFA – Google auth	3,67	5. – 6.
MFA – Email	3,34	7. – 9.
Biometrie	3,67	5. – 6.

7.2 Celkové shrnutí

Jako nejlepší metoda dle zadaných kritérií vyšla metoda přihlášení jménem a heslem. Neměla významné nedostatky v žádném z kritérií a je solidním způsobem autentizace ve všech situacích. Metoda je dostatečně zabezpečená, je jednoduché ji používat i implementovat. Její umístění je také možné připsat faktu, že se jedná o primární způsob autentizace v Mendixu, kvůli čemuž se na ní poskytovatelé nejvíce soustředí.

Její hlavními nevýhodami je složitost případné modifikace, tedy pokud by bylo nutné provést úpravy na úrovni Javy. Zde je nutné rozumět logice Mendix přihlašování a zároveň adekvátně ošetřit nově rizika a zranitelnosti, která mohou vzniknout.

Na druhém místě se umístila metoda Mendix SSO. Ačkoliv dosáhla o 0,33 vyššího skóre jak OIDC SSO, dle autorova názoru jsou na podobné úrovni. Každá pokrývá jiné scénáře použití, které nejsou v hodnocení zahrnuty. Mendix SSO dokáže pracovat jen s Mendix účty, což výrazně omezuje reálnou využitelnost v praxi a aktuálně jde spíše o prostředek autentizace pro specializované Mendix pracovníky. Kromě této vlastnosti jde avšak o použitelnou a kvalitní metodu autentizace, u které je možné v blízké době očekávat velké zlepšení po vydání Mendix „Build your own IdP“ (Mendix 2022t). Pokud tato BYOIDP funkce naplní své sliby, je možné očekávat, že se stane primární a nejpoužívanější autentizační metodou.

Třetí a čtvrté místo je sdíleno OIDC SSO a metodou jméno, heslo & Captcha.

OIDC SSO je kvalitní modul. Z pohledu vývojáře je ocenitelný primárně jeho způsob implementace, kdy není upravena defaultní login akce jako např. u LDAP či OTP. Modul je tak možné používat i s jinými metodami a nedochází ke konfliktům. Jediná kritéria ve kterých neuspěl vyplývala z charakteru SSO. Z pohledu funkčnosti a bezpečnosti obsadilo OIDC SSO s několika dalšími metodami první místo. Funkčně jde o modul umožňující práci s několika různými SSO IdP zároveň, přičemž dokáže zajistit spolehlivý a bezpečný chod autentizací.

Jméno, heslo & Captcha je jednoduché rozšíření základní login akce, kde není nutná přílišná úprava kódu a je vhodným řešením pro provozovatele aplikací, kteří chtějí částečně zvýšit bezpečnost přihlašování. V prostředí Mendixu je možné jej implementovat za pomoci jednoduchého widgetu. Obecně se nejedná o velkou změnu v autentizaci. Hodnocení tomuto faktu odpovídá.

Páté a šesté místo je opět sdíleno, a to více faktorovou Google Authenticator a biometrickou autentizací. MFA pomocí Google Authenticator je jednoduché, spolehlivé a bezpečné řešení více faktorové autentizace. Jediným problémem je nutnost využívat externí mobilní zařízení s aplikací, což může být nepohodlné pro uživatele. I přes to se ale jedná dle autorova názoru o jedno nejvhodnější MFA řešení této práce. Struktura modulu umožňuje implementovat procedury dle potřeb provozovatele a zároveň není zasahováno do defaultních login funkcí aplikace. Modul je obecně velmi snadné používat a je možné jej i implementovat tak, aby například byl použit jen při přihlašování administrátorů či jiných, specifikovatelných skupin.

Biometrická autentizace z Mendix modulu Native Mobile Resources není příliš vhodná pro používání jako autentizační prostředek. Ačkoliv se na výsledném hodnocení neumístil nejhůře, je možné ji považovat za nejhorší řešení z pohledu celkové použitelnosti. Situace, kdy je vhodné použít biometrické akce modulu, jsou dosti omezené a vhodnější by bylo tento modul používat jako druhý faktor či pouze jako kontrolu při provádění důležitějších operací v aplikaci. Jako autentizační prostředek není implementace tohoto modulu nijak intuitivní, pohodlná či bezpečná.

Na třech posledních místech se umístily metody LDAP, MFA – Email a OTP.

MFA – Email je metoda kvalitně splňující účel, pro který byla vytvořena. Modul je avšak vhodný jen pro situace, kdy aplikace potřebuje pouze tento způsob přihlášení. Modul modifikuje defaultní login akci, což komplikuje využívání dalších přihlašovacích metod. Jeho velkou výhodou je vysoká konfigurovatelnost, kdy je možné mnoho vlastností nastavit v modulových konstantách a jeho celková funkčnost je programovatelná ve třech specifických MF, díky čemuž není nutné upravovat Java kód. Zároveň se jedná o jediný z analyzovaných modulů, který obsahuje unit testy. Tento modul je autorem vnímán jako velmi kvalitní a jeho neúspěšné umístění je možné vnímat jako charakter použité metody, kdy je k provozu nutné pracovat s emailovým hostingem. I pro uživatele se nejedná o nejpohodlnější metodu na používání. Ovšem je velmi vhodný pro situace, kdy je nutné pro uživatele, či jen určitou skupinu uživatelů, přidat dodatečný faktor autentizace.

LDAP modul není dostatečný pro používání v současných podmínkách, ačkoliv by uživatel při používání nepoznal rozdíl od některých SSO metod. Modul není aktualizovaný, udržovaný či dostatečně dokumentovaný. Pro verzi MX9 není ani podporován. Práce s ním je obtížná, některé změny se musí provádět i na úrovni Javy, kterou je nutné i debuggovat a kontrolovat při problémech, které při konfiguraci mohou nastat. Dle autorova názoru je tato pozice LDAP modulu patří. Pokud provozovatel není nucen LDAP využívat, poté by bylo vhodnější se mu vyhnout a implementovat jiné řešení.

OTP modul vytvořený v rámci této práce nezískal dobrá hodnocení oproti ostatním metodám. Opět je zde ale hlavní limitací charakter samotné metody, ne jeho implementace, kdy primárními problémy je nízká pohodlnost pro uživatele a nutnost nastavit si emailový server pro provozovatele. Implementace modulu je provedena modifikací standardní login akce, kde byla snaha zachovat jak přihlašování jménem a heslem, tak i všechna hlavní bezpečnostní opatření. Pro reálné využití tohoto, a dalších OTP modulů, by bylo vhodné, pokud by Mendix umožnil přihlašování jiným způsobem, jak jménem a heslem již defaultně

v aplikaci, nebo pokud by bylo možné přidávat nové způsoby přihlášení bez úpravy defaultního přihlašování.

8 Doporučení

8.1 Pro firemní ERP

Při tvorbě firemního ERP systému v Mendixu by bylo vhodné zvážit použití SSO autentizace či kterékoliv dvou faktorové autentizace. Je možné se domnívat že provozovatel, tedy firma, bude mít vyšší bezpečnostní požadavky a proto pro něj bude mít vyšší váhu dimenze bezpečnosti. ERP a další firemní systémy pracují s citlivými, a pro podnikání velmi důležitými informacemi, které je potřeba chránit.

Ačkoliv tyto autentizační metody patří mezi dražší a náročnější na implementaci, v případě firemních systémů je časté, že společnosti pracují s centrální správou uživatelů, se kterou následně půjde i tato aplikace propojit. To je velká výhoda pro SSO, kdy je možné využít stejné firemní uživatelské účty napříč vícero firemními aplikacemi, díky čemuž se ceny provozování těchto řešení rozloží mezi ně. Zároveň v případě firemních systémů se nepracuje s tolika uživateli, jako v ostatních případech a dále aplikace nepracuje s žádnými anonymními uživateli.

Z pohledu uživatelů může být nepohodlné používat SSO či více faktorovou autentizaci, avšak nejspíše pro ně bude pochopitelné, proč byl tento způsob zvolen a že se jedná o důležitá data a aplikace. Díky tomu nejspíše nebude jejich používání problematické a ze strany uživatelů nevznikne velký odpor.

8.2 Pro E-shop

V případě Mendix E-shop aplikace by bylo vhodné použít SSO či standardní jméno a heslo. V případě, že by aplikace ukládala nejen osobní údaje, ale i platební metody, jako např. kreditní karty, poté by v určitých případech bylo na místě implementovat i více faktorovou autentizaci.

Standardně je cílem provozovatele usnadnit nákupy v prostředí aplikace, s čímž souvisí např. ukládání adresy pro doručení či kontaktních údajů pro další nákupy či ukládání účtenek a usnadnění komunikace. V tomto ohledu jsou vhodné OIDC SSO autentizační metody, kdy se uživatel může do aplikace registrovat i přihlásit za pomocí svých existujících účtů, např. u Google, Facebook, či jiných služeb. Za předpokladu, že uživatel se přihlašuje ze zařízení, kde již na těchto službách přihlášen je, bude pro něj přihlášení rychlejší a pohodlnější, než pokud by musel projít standardní registrací. Možností by ovšem mělo být i standardní přihlášení jménem a heslem.

Dle situace by bylo na místě zvážit i dvou faktorovou autentizaci. Zde by nejspíše byla nejvhodnější varianta pomocí OTP kódu přes email či SMS. Nemělo by se ovšem jednat o standard, ale jen doplněk ve výjimečných případech a situacích. Může jít například o situaci úpravy profilu či úpravě nebo použití samotných platebních metod. Vybrané

autentizační metody by ovšem měly být použitelné jak na desktopových, tak i na mobilních a tabletových zařízeních.

8.3 Pro zpravodajský web

Pro zpravodajské weby by měla být dostačující autentizace jménem a heslem, není zde potřeba budovat a používat složitější řešení. V ojedinělých případech je možné zvážit opět používání OIDC SSO řešení, kdy se usnadní mnoha uživatelům registrace i přihlašování, avšak oproti předchozím dvěma aplikacím není pro provozovatele zpravodajského webu natolik důležité, aby byl uživatel přihlášen. Za předpokladu, že provozovateli nejvíce záleží na tom, aby se měl co nejvyšší návštěvnost, jde spíše o zajištění co nejvyššího množství anonymních uživatelů, než o velký počet registrovaných uživatelů.

Záleží ovšem na výhodách přihlášených uživatelů. Pokud uživatel může po přihlášení zobrazit obsah, který by nebyl dostupný nepřihlášenému uživateli, poté se spíše uživatelé budou registrovat, než v případě možnosti odebírání newsletteru či možnosti komentování příspěvků.

V každém případě je nutné, aby zvolené metody patřily mezi pohodlnější, aby uživatel nepřerušil registraci či přihlášení kvůli vysoké komplexitě. Zároveň je nutné, aby je bylo možné využít na všech zařízeních, a to primárně na zařízeních mobilních.

9 Ověření výsledků práce

V následujících podkapitolách jsou interpretovány nejdůležitější poznatky z validací od odborníků z praxe. Celé, neupravené odpovědi je možné nalézt v Příloha C: Odpovědi na Otázky.

9.1 Odborník A

První odborník hodnotí práci pozitivně, nalezl ale několik nedostatků. Prvním z nich je, že při testování a vyhodnocování metod se pracovalo s kontextem jedné aplikace. Zde je zdůrazňováno, že v praxi se pracuje s desítkami aplikací, kde je ve výsledku snazší pracovat s SSO řešeními. Je pravda že tento fakt přímo v kritériích obsažen není.

Dalším nedostatkem je stejná váha dimenzí, kdy hlavním cílem autentizace je bezpečnost, ne snadnost používání. Proto měla mít dimenze bezpečnosti vyšší váhu na celkové hodnocení.

S doporučeními odborník souhlasí, ale má 2 výhrady k doporučením pro firemní ERP systém. Zmiňuje, že není pravda že by firemní ERP systémy nemusely pracovat s anonymními uživateli a dále doplňuje, že pro situace, kdy uživatelé používají (a administrátoři spravují) stovky aplikací, je SSO, MFA, či OTP lepší a pohodlnější metodou jak jméno a heslo.

Dále zmiňuje, že mohla být analyzována i metoda SSO pomocí SAML 2.0. Tato metoda nebyla zahrnuta jelikož SSO je v práci pokryto 2 jinými moduly.

Kromě těchto nálezů považuje práci za kvalitní, dobře zpracovanou a detailní. Nebyly nalezeny žádné nepravdivé výroky či chyby.

9.2 Odborník B

Druhý odborník primárně zmiňuje, že hodnotící dimenze mohly obsahovat kritéria pro hodnocení Compliance, schopnost integrace s jinými systémy, flexibilitu a dependence na jiné moduly třetích stran.

Dále zmiňuje, že i přes vysoké umístění autentizační metody jména a hesla, v jejich prostředí není doporučené tyto metody používat. Důvodem je nízká bezpečnost, způsobená slabými uživatelskými hesly.

Nakonec ještě zmiňuje, že aplikace jako e-shopy a zpravodajské weby nejsou typické use-case pro Mendix aplikace. Autor se měl proto více zaměřit na interní firemní aplikace.

9.3 Odborník C

Třetí odborník obecně souhlasí s výstupy práce a stejně jako ostatní oborníci zdůrazňuje fakt, že v praxi je vhodné používat centrální SSO řešení. Dále komentuje, že při používaní metod jména a hesla v aplikacích je nutná vysoká důvěra administrátorům a vývojářům, že je vše nastaveno a naimplementováno správně. SSO umožňuje snazší správu uživatelů.

S vybranými metodami souhlasí, stejně tak s dimenzemi a postupem hodnocení.

Práci hodnotí jako velmi přínosnou a dále navrhuje další oblast, kterou by bylo možné analyzovat – šifrování dat uvnitř aplikace a bezpečné ukládání dat.

9.4 Odborník D

Dle posledního odborníka jsou v práci pokryty všechny dostupné autentizační metody. Opět je zde ale vyčteno chybějící kritérium pro benefity centralizovaného managementu uživatelů. Argumentuje, že zde nejsou pokryty situace, kdy jedno přihlášení umožní uživateli přístup do mnoha aplikací najednou.

Dále je zde kritizováno testování dimenze pohodlnosti, kdy není brán v potaz fakt, že při používání SSO si uživatel musí pamatovat pouze jedno heslo. V případě vícero aplikací a vícero hesel (pokud nepoužívá všude stejná), si musí pamatovat hesel mnoho, což může být zmatečné a často může dojít k situacím že heslo zapomene. Tím by SSO metody měly být pohodlnější jak přihlašování jménem a heslem.

S doporučeními pro jednotlivé aplikace odborník souhlasí. Práce je dle něj detailní a přínosná, a to primárně svým náhledem do interního fungování autentizačních metod.

9.5 Autorův komentář k validacím

Všechny validace jsou velmi přínosné a každá z nich zmiňuje důležité poznatky. Pokud by se analýza metod prováděla opětovně, určitě by bylo vhodné zapracovat zmíněné změny v dimenzích a kritériích. Dále by bylo dále vhodné provádět analýzu v kontextu používání napříč vícero aplikacemi, kde by lépe vyšly metody jako SSO či LDAP, oproti přihlašování jménem a heslem.

Obecně je obtížné sestavit mechanismus pro hodnocení autentizačních metod. Dle autorova názoru není finální podoba hodnocení metod v této práci špatná, což se potvrdilo i u odborníků, je zde ovšem samozřejmě prostor pro zlepšení.

Odborníci pozitivně hodnotili celkovou kvalitu, úroveň detailu a přínosnost práce. Zároveň souhlasili až na pár dodatků s vytvořenými doporučeními.

Závěr

Shrnutí výsledků práce

Platforma Mendix poskytuje mnoho způsobů autentizace, některé defaultně, jiné v externích modulech. Tato práce se zabývala analýzou autentizačních metod, které byly dále hodnoceny ze tří dimenzí – Použitelnosti, bezpečnosti a nákladů a znalostí. Dle výsledků z kapitol 6 a 7 vyplývá, že obecně nejvhodnějšími autentizačními metodami v prostředí Mendix jsou metody jména a hesla a SSO. Dle výsledků se na prvním místě umístila metoda jména a hesla, což je primárním autentizačním prostředkem v Mendixu. Tento výsledek je avšak obecný, a dosti závislý na kontextu použití.

V rámci kapitoly 8 jsou proto představeny 3 různé typy aplikací a pro každou z nich jsou doporučeny nejvhodnější autentizační metody. Pro firemní ERP systémy je nejvhodnější používat centralizované SSO řešení, případně MFA řešení. Toto doporučení se podařilo potvrdit i při validaci několika odborníky, kteří s těmito aplikacemi pracují. Pro e-shop je výhodné také používat SSO, ale oproti firemním aplikacím, zde je lepší používat OIDC SSO, kde se uživatel může registrovat a přihlašovat pomocí svých účtů na sociálních sítích, např Google, Facebook atd. A v posledním případě, tj. zpravodajském webu, je nejvhodnější uvažovat o klasickém jménu a heslu SSO jako v případě e-shopu.

Všechny analyzované metody byly implementovány v demo aplikaci, dostupné s touto prací pro kohokoliv, kdo by potřeboval více detailní, technický pohled.

Platforma Mendix se stále vyvíjí a spolu s ní se mění i samotné autentizační metody. V budoucnu lze očekávat, že po vydání Mendix BYOIDP, viz 5.1.4, se hodnocení jednotlivých metod změní, kdy Mendix SSO metoda bude značně rozšířena o nové funkčnosti a stane se primárním autentizační metodou na úkor jména a hesla.

Míra splnění cílů práce

Hlavním cílem práce bylo vymezit, analyzovat, vyhodnotit z různých pohledů/dimenzí a následně srovnat možnosti autentizace dostupné v aktuální verzi Low-Codové platformy Mendix.

Dílčími cíli práce bylo:

- Doporučit nejvhodnější metody autentizace pro aplikace různých velikostí, potřeb a účelů.
- Vytvoření demo aplikace využívající rozebírané autentizační metody

Všechny tyto cíle se podařilo pokrýt a splnit. Vybrané metody jsou vypsány v kapitole 2.1, dimenze a proces hodnocení popsán v kapitole 2.2 a samotná analýza je obsažena v kapitole 5. Finální porovnání a zanesení výsledků je možné nalézt v kapitolách 6 a 7.

Doporučení se nachází v kapitole 8 a primárně vychází z poznatků z kapitol 5, 6 a 7.

Demo aplikace byla vypracována autorem práce během tvorby této práce a je dostupná ke stažení spolu s touto prací. Jak v ní byly implementovány jednotlivé metody je obsaženo v kapitole 5 a celkový popis je dostupný v Příloha A:Demo Aplikace.

Pro ověření správnosti a přínosnosti práce bylo provedeno ověření se 4 odborníky z praxe, které je obsaženo v kapitole 9.

Využitelnost a přínos práce

Nejvyšším přínosem práce je samotný průzkum a analýza autentizačních metod. Poznatky je možné využít při dalších analýzách či pro samotné implementace či modifikace jednotlivých metod. Pro uživatele a vývojáře platformy Mendix je jistě přínosná i možnost prohlédnutí demo aplikace a způsoby implementace jednotlivých modulů. Dále mohou být dobrým zdrojem informací pro kohokoliv, kdo chce nabýt nových znalostí v oblasti autentizace v Mendixu.

Přínosná jsou ovšem také vytvořená doporučení a celkové hodnocení metod, kde tyto výsledky mohou být využity provozovateli aplikací. Zde je ale nutné dodat, že celkové hodnocení a pořadí metod může být jiné pro různé kontexty či prostředí, jak bylo zmiňováno ve validacích. Z tohoto pohledu může práce pomoci k vyšší bezpečnosti a obecně kvalitnějšímu výběru autentizačních metod.

Déle lze považovat za přínosnou vytvořenou metodu pro hodnocení autentizačních metod, na kterou je možné navázat, doplnit, či se inspirovat při dalších hodnoceních.

Další náměty pro řešení

Během tvorby práce vyvstaly tyto náměty na další práce v této oblasti:

- **Tvorba formalizovaného modelu/metody pro hodnocení autentizačních metod**
    - Při rešerši a průzkumu dostupných metod pro hodnocení autentizačních metod bylo obtížné najít obecný model/obecná metoda. Nalezené metody byly často moc konkretizované či jinak nevhodné pro aktuální případ.
- **Nákladová analýza autentizačních metod v Mendixu**
    - Při hodnocení metod bylo původně cílem zahrnout i detailní hodnocení finanční stránky, jako např pořizovací náklady, provozní náklady apod. Hodnocení těchto aspektů se ovšem projevilo jako dosti komplexní a obtížně řešitelný problém, jelikož je tato oblast ovlivněna mnoha faktory.

- **Studie uživatelského postoje k používání jednotlivých metod**
    - Pro hodnocení uživatelského mínění či postoje k jednotlivým autentizačním metodám neexistuje mnoho literatury či akademických prací, ačkoliv výsledky by mohly být přínosné. Při rešerši byly nalezeny spíše teoretické práce, než práce pracující s konkrétními daty a výsledky.

Použitá literatura

ACUMEN RESEARCH AND CONSULTING, 2022. Low-Code Development Platform Market Size Was Valued at USD 16 Billion in 2021 and Will Achieve USD 159 Billion by 2030 growing at 28.8% CAGR Owing to the Growing Need to Accelerate Digital Transformation-Exclusive Report by Acumen Research and Consulting. *GlobeNewswire News Room* [online] [vid. 2022-10-14]. Dostupné z: https://www.globenewswire.com/news-release/2022/07/31/2489071/0/en/Low-Code-Development-Platform-Market-Size-Was-Valued-at-USD-16-Billion-in-2021-and-Will-Achieve-USD-159-Billion-by-2030-growing-at-28-8-CAGR-Owing-to-the-Growing-Need-to-Accelerate.html

ANDRIOAIE, Andra, 2022. Authentication vs Authorization: What's the Difference? *Heimdal Security Blog* [online]. [vid. 2022-10-11]. Dostupné z: https://heimdalsecurity.com/blog/authentication-vs-authorization/

APPIAN, 2021. *What Are The Benefits of Low-Code | Low-Code Basics* [online] [vid. 2022-10-14]. Dostupné z: https://appian.com/low-code-basics/benefits.html

APPRONTO, 2019. *Mendix Marketplace - SignIn microflow for Mx7 / Mx8* [online] [vid. 2023-01-03]. Dostupné z: https://marketplace.mendix.com/link/component/66443

ARIAS, Dan, 2021. Hashing in Action: Understanding bcrypt. *Auth0 - Blog* [online] [vid. 2022-11-03]. Dostupné z: https://auth0.com/blog/hashing-in-action-understanding-bcrypt/

ARIS, 2011. *Event-driven process chain (EPC)* [online] [vid. 2022-10-12]. Dostupné z: https://www.ariscommunity.com/event-driven-process-chain

AVERTRA, 2022. Build a Secure Biometric Authentication with Mendix Native Mobile. *Avertra* [online]. [vid. 2023-02-15]. Dostupné z: https://avertra.com/build-a-secure-biometric-authentication-with-mendix-native-mobile/

BABIČ, Peter, 2020. Are OTP secrets stored in plaintext. *Peter Babič* [online] [vid. 2022-12-04]. Dostupné z: https://peterbabic.dev/blog/are-otp-secrets-stored-plaintext

BOONKRONG, Sirapat, 2020. *Authentication and Access Control: Practical Cryptography Methods and Tools*. 1. edice. New York, NY: Apress. ISBN 978-1-4842-6570-3.

BRANDESSENCE a PR NEWSWIRE, 2021. *At +26.1% CAGR, Low-code Development Platform Market size is Expected to reach 65.15 Billion by 2027, says Brandessence Market Research* [online] [vid. 2022-10-14]. Dostupné z: https://www.prnewswire.com/news-releases/at-26-1-cagr-low-code-development-platform-market-size-is-expected-to-reach-65-15-billion-by-2027--says-brandessence-market-research-301243189.html

BROWN, Schuyler, 2022. *The Difference Between SAML vs OIDC | StrongDM* [online] [vid. 2022-11-15]. Dostupné z: https://www.strongdm.com/blog/oidc-vs-saml

CREATIO, 2021. How much faster is low-code development comparing to traditional development? *Statista* [online] [vid. 2022-10-14]. Dostupné z: https://www-statista-com.zdroje.vse.cz/statistics/1254662/low-code-development-speed-compared-traditional-it/

CWE, 2022. *CWE - CWE-656: Reliance on Security Through Obscurity (4.9)* [online] [vid. 2022-12-12]. Dostupné z: https://cwe.mitre.org/data/definitions/656.html

DE GELDER, Chris, 2021. *Mendix Forum - Question Details* [online] [vid. 2023-02-01]. Dostupné z: https://forum.mendix.com/link/space/integrations/questions/109928

DINH, Kathy, 2019. How to implement secure Biometric Authentication on mobile devices. *freeCodeCamp.org* [online] [vid. 2023-02-15]. Dostupné z: https://www.freecodecamp.org/news/how-to-implement-secure-biometric-authentication-on-mobile-devices-4dc518558c5c/

EUROPEAN COMMISION, 2014. *Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC* [online]. 23. červenec 2014. [vid. 2022-10-09]. Dostupné z: http://data.europa.eu/eli/reg/2014/910/oj/eng

EUROPEAN COMMISION, 2022. *eIDAS Levels of Assurance* [online] [vid. 2022-10-09]. Dostupné z: https://ec.europa.eu/digital-building-blocks/wikis/digital-building-blocks/wikis/display/DIGITAL/eIDAS+Levels+of+Assurance

FOXPASS, 2023. *Foxpass - Pricing* [online] [vid. 2023-03-01]. Dostupné z: https://www.foxpass.com/pricing

GOOGLE, 2023. OpenID Connect | Authentication. *Google Developers* [online] [vid. 2023-01-25]. Dostupné z: https://developers.google.com/identity/openid-connect/openid-connect

GOOGLE CLOUD, 2023a. Pricing | Identity Platform. *Google Cloud* [online] [vid. 2023-03-01]. Dostupné z: https://cloud.google.com/identity-platform/pricing

GOOGLE CLOUD, 2023b. Pricing | reCAPTCHA Enterprise. *Google Cloud* [online] [vid. 2023-03-01]. Dostupné z: https://cloud.google.com/recaptcha-enterprise/pricing

GRASSI, Paul A, Michael E GARCIA a James L FENTON, 2017. *Digital identity guidelines: revision 3* [online]. NIST SP 800-63-3. Gaithersburg, MD: National Institute of Standards and Technology [vid. 2022-10-09]. Dostupné z: doi:10.6028/NIST.SP.800-63-3

GROOT, Bart, 2014. *Doing custom stuff when a user logs in | BartGroot.nl* [online] [vid. 2023-01-03]. Dostupné z: https://bartgroot.nl/mendix/custom-checks-on-login/

GUEVARA, Holly, 2021. What is SAML and how does SAML Authentication Work. *Auth0 - Blog* [online] [vid. 2022-11-12]. Dostupné z: https://auth0.com/blog/how-saml-authentication-works/

HELKALA, Kirsi a Einar SNEKKENES, 2009. Formalizing the ranking of authentication products. *Information Management & Computer Security* [online]. **17**(1), 30–43. ISSN 0968-5227. Dostupné z: doi:10.1108/09685220910944740

HICKEY, James, 2021. Why use a standardized auth protocol? *FusionAuth* [online] [vid. 2022-12-04]. Dostupné z: https://fusionauth.io/blog/2021/06/10/why-consider-standards-based-auth-options-excerpt

IBEAKANMA, Chioma, 2022. What Is Peppering in Password Security and How Does It Work? *MUO* [online] [vid. 2022-11-03]. Dostupné z: https://www.makeuseof.com/what-is-peppering-how-does-it-work/

IBM, 2022. *IBM Documentation* [online] [vid. 2022-10-09]. Dostupné z: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/en/ibm-mq/9.0?topic=mechanisms-identification-authentication

INTERESSE, Giulia, 2022. The Low-Code/No-Code Market in China and its Huge Growth Potential. *China Briefing News* [online] [vid. 2022-10-15]. Dostupné z: https://www.china-briefing.com/news/low-code-no-code-industry-in-china-market-scope-opportunities-for-foreign-investors/

INTREXX, 2022. *LowCode Soulution - INTREXX* [online] [vid. 2022-10-15]. Dostupné z: http://www.lowcode.cz/

ISDECISIONS, 2021. MFA vs. 2FA vs. 2SV: How to Choose the Right Multi-Factor Authentication. *IS Decisions* [online]. B.m.: IS Decisions [vid. 2022-10-10]. Dostupné z: https://www.isdecisions.com/mfa-vs-2fa-vs-2step-how-to-choose-the-right-multi-factor-authentication/

JETVEO, 2022. *O nás | Jetveo* [online] [vid. 2022-10-15]. Dostupné z: https://jetveo.io/cs/firma

JUMPCLOUD, 2023a. JumpCloud: SSO and Active Directory Reimagined. *JumpCloud* [online] [vid. 2023-02-02]. Dostupné z: https://jumpcloud.com/

JUMPCLOUD, 2023b. Pricing. *JumpCloud* [online]. [vid. 2023-03-01]. Dostupné z: https://jumpcloud.com/pricing

KADHOM, Naoufal, 2018. *React Native Touch ID* [online]. Java, JS. 16. červen 2018. [vid. 2023-02-13]. Dostupné z: https://github.com/naoufal/react-native-touch-id

KHAN, Rahim, 2017. Modern Authentication Techniques in Smart Phones: Security and Usability Perspective. *International Journal of Advanced Computer Science and Applications* [online]. **8**, 10. Dostupné z: doi:10.14569/IJACSA.2017.080142

KUČERA, Jakub, 2012. *Perspektivní bezpečnostní metody pro přímé bankovní kanály* [online]. Plzeň [vid. 2022-10-30]. Diplomová práce. Západočeská univerzita v Plzni. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/3084/1/DP_JKucera.pdf

KUMAR, Vijay, 2022. *OWASP-Single-Sign-On-Vijay.pdf* [online] [vid. 2022-11-08]. Dostupné z: https://owasp.org/www-pdf-archive//OWASP-Single-Sign-On-Vijay.pdf

LARDINOIS, Simon, 2021. A closer look at the security of React Native biometric libraries. *NVISO Labs* [online] [vid. 2023-02-15]. Dostupné z: https://blog.nviso.eu/2021/04/06/a-closer-look-at-the-security-of-react-native-biometric-libraries/

LAVENDER, Greg a Mark WAHL, 2003. INTERNET DIRECTORY SERVICES USING THE LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL. 30.

LDAP, 2018. Learn About LDAP. *LDAP.com* [online] [vid. 2022-11-17]. Dostupné z: https://ldap.com/learn-about-ldap/

LEBLANC, Jonathan a Tim MESSERSCHMIDT, 2016. *Identity and Data Security for Web Development: Best Practices*. 1. edice. Sebastopol, CA: O'Reilly Media. ISBN 978-1-4919-3694-8.

LXA, 2022. *Low code / no code Best Practice Guide* [online] [vid. 2022-10-13]. Dostupné z: https://www.lxahub.com/ebook/low-code-no-code-best-practice-guide

MAIYA, Arun S, 2022. ktrain: A Low-Code Library for Augmented Machine Learning. 6.

MATHUR, Vrinda, 2022. *All you need to know about OTP Authentication Method | Analytics Steps* [online] [vid. 2023-01-15]. Dostupné z: https://www.analyticssteps.com/blogs/all-you-need-know-about-otp-authentication-method

MEARIAN, Lucas, 2022. Low-code tools can fill a void caused by the Great Resignation. *Computerworld* [online] [vid. 2022-10-14]. Dostupné z: https://www.computerworld.com/article/3658908/how-low-code-tools-are-filling-a-void-caused-by-the-great-resignation.html

MENDIX, 2020. *Mendix Marketplace - Mx Model reflection* [online] [vid. 2023-01-27]. Dostupné z: https://marketplace.mendix.com/link/component/69

MENDIX, 2022a. *Advanced Custom Settings in Mendix Runtime* [online] [vid. 2023-01-07]. Dostupné z: https://docs.mendix.com/refguide/tricky-custom-runtime-settings/

MENDIX, 2022b. *App Settings* [online] [vid. 2022-10-21]. Dostupné z: https://docs.mendix.com/refguide/app-settings/

MENDIX, 2022c. Cloud Architecture. *Mendix* [online] [vid. 2022-10-29]. Dostupné z: https://www.mendix.com/evaluation-guide/enterprise-capabilities/cloud-architecture/

MENDIX, 2022d. *Clustered Mendix Runtime* [online] [vid. 2023-01-07]. Dostupné z: https://docs.mendix.com/refguide/clustered-mendix-runtime/

MENDIX, 2022e. Collaboration Channels. *Mendix* [online] [vid. 2022-10-24]. Dostupné z: https://www.mendix.com/evaluation-guide/app-lifecycle/collaboration-channels/

MENDIX, 2022f. *Community Commons* [online] [vid. 2023-01-06]. Dostupné z: https://docs.mendix.com/appstore/modules/community-commons-function-library/

MENDIX, 2022g. Identity and Access Management (IAM). *Mendix* [online] [vid. 2022-12-28]. Dostupné z: https://www.mendix.com/evaluation-guide/enterprise-capabilities/identity-and-access-management-iam/

MENDIX, 2022h. *Implement Best Practices for App Security* [online] [vid. 2023-01-06]. Dostupné z: https://docs.mendix.com/howto/security/best-practices-security/

MENDIX, 2022i. Low-Code Guide. *Mendix* [online] [vid. 2022-10-12]. Dostupné z: https://www.mendix.com/low-code-guide/

MENDIX, 2022j. *Mendix Marketplace - Category Authentication* [online] [vid. 2022-12-28]. Dostupné z: https://marketplace.mendix.com/link/tag/41

MENDIX, 2022k. *Mendix Marketplace - Community Commons* [online] [vid. 2023-01-27]. Dostupné z: https://marketplace.mendix.com/link/component/170

MENDIX, 2022l. *Mendix Marketplace - Encryption* [online] [vid. 2023-01-27]. Dostupné z: https://marketplace.mendix.com/link/component/1011

MENDIX, 2022m. *Mendix Marketplace - Nanoflow Commons* [online] [vid. 2023-01-27]. Dostupné z: https://marketplace.mendix.com/link/component/109515

MENDIX, 2022n. *Mendix Marketplace - Native Mobile Resources* [online] [vid. 2023-01-27]. Dostupné z: https://marketplace.mendix.com/link/component/109513

MENDIX, 2022o. *Mendix Single Sign-On* [online] [vid. 2023-01-17]. Dostupné z: https://docs.mendix.com/developerportal/deploy/mendix-sso/

MENDIX, 2022p. *Microflows* [online] [vid. 2022-10-18]. Dostupné z: https://docs.mendix.com/refguide/microflows/

MENDIX, 2022q. *Nanoflows* [online] [vid. 2022-10-18]. Dostupné z: https://docs.mendix.com/refguide/nanoflows/

MENDIX, 2022r. *Pages* [online] [vid. 2022-10-19]. Dostupné z: https://docs.mendix.com/refguide/pages/

MENDIX, 2022s. Security. *Mendix* [online] [vid. 2022-10-25]. Dostupné z: https://www.mendix.com/evaluation-guide/enterprise-capabilities/security/

MENDIX, 2022t. *Set Up an SSO (BYOIDP)* [online] [vid. 2023-01-18]. Dostupné z: https://docs.mendix.com/developerportal/control-center/set-up-sso-byoidp/

MENDIX, 2022u. *Studio Pro 9.11* [online] [vid. 2023-01-06]. Dostupné z: https://docs.mendix.com/releasenotes/studio-pro/9.11/

MENDIX, 2022v. Why Was Mendix Founded? *Mendix* [online] [vid. 2022-10-16]. Dostupné z: https://www.mendix.com/evaluation-guide/why-founded/

MENDIX, 2023a. *Apidocs-Core* [online] [vid. 2023-01-03]. Dostupné z: https://apidocs.rnd.mendix.com/9/runtime/com/mendix/core/Core.html

MENDIX, 2023b. *Email Connector* [online] [vid. 2023-02-09]. Dostupné z: https://docs.mendix.com/appstore/connectors/email-connector/

MENDIX, 2023c. *Login Behavior* [online] [vid. 2023-01-04]. Dostupné z: https://docs.mendix.com/refguide/login-behavior/

MENDIX, 2023d. *Mendix - License Purchase* [online] [vid. 2023-03-01]. Dostupné z: https://mxbpconfig.mendixcloud.com/index.html

MENDIX, 2023e. Mendix Basic Package. *Mendix* [online] [vid. 2023-03-01]. Dostupné z: https://www.mendix.com/pricing/basic-package/

MENDIX, 2023f. *Mendix Forum - Community Feed* [online] [vid. 2023-01-12]. Dostupné z: https://forum.mendix.com/p/community

MENDIX, 2023g. *OIDC SSO* [online] [vid. 2023-01-27]. Dostupné z: https://docs.mendix.com/appstore/modules/oidc/

MENDIX RESEARCH & DEVELOPMENT, 2022. *Make It Native 9 - Apps on Google Play* [online] [vid. 2023-02-13]. Dostupné z: https://play.google.com/store/apps/details?id=com.mendix.developerapp.mx9&hl=en_US&gl=US

MENDIX SUPPORT TEAM, 2023. *Default Login Action*. 2. leden 2023. Osobní komunikace.

MICROSOFT, 2022a. *Add password-based single sign-on to an application - Microsoft Entra* [online] [vid. 2022-11-12]. Dostupné z: https://learn.microsoft.com/en-us/azure/active-directory/manage-apps/configure-password-single-sign-on-non-gallery-applications

MICROSOFT, 2022b. *Azure single sign-on SAML protocol - Microsoft Entra* [online] [vid. 2022-11-12]. Dostupné z: https://learn.microsoft.com/en-us/azure/active-directory/develop/single-sign-on-saml-protocol

MICROSOFT, 2022c. *Plan a single sign-on deployment - Microsoft Entra* [online] [vid. 2022-11-12]. Dostupné z: https://learn.microsoft.com/en-us/azure/active-directory/manage-apps/plan-sso-deployment

MICROSOFT, 2022d. *What is Low-Code Development? | Microsoft Power Apps* [online] [vid. 2022-10-12]. Dostupné z: https://powerapps.microsoft.com/en-us/what-is-low-code/

MICROSOFT, 2022e. *What is: Multifactor Authentication* [online] [vid. 2022-10-10]. Dostupné z: https://support.microsoft.com/en-us/topic/what-is-multifactor-authentication-e5e39437-121c-be60-d123-eda06bddf661

MICROSOFT, 2022f. *What is single sign-on in Azure Active Directory?* [online] [vid. 2022-11-08]. Dostupné z: https://learn.microsoft.com/en-us/azure/active-directory/manage-apps/what-is-single-sign-on

MOTA, Tiago, 2020. OutSystems 101: Splitting the flow! *Medium* [online] [vid. 2022-10-12]. Dostupné z: https://itnext.io/outsystems-101-splitting-the-flow-c2a0afb0e60f

M'RAIHI, David, Johan RYDELL, Mingliang PEI a Salah MACHANI, 2011. *TOTP: Time-Based One-Time Password Algorithm* [online]. Request for Comments RFC 6238. B.m.: Internet Engineering Task Force [vid. 2023-02-07]. Dostupné z: doi:10.17487/RFC6238

MVČR, 2020. *Ministerstvo vnitra zveřejňuje dokument konkretizující minimální požadavky na kvalifikované eID systémy a eID prostředky - Ministerstvo vnitra České republiky* [online] [vid. 2022-10-04]. Dostupné z: https://www.mvcr.cz/clanek/ministerstvo-vnitra-zverejnuje-dokument-konkretizujici-minimalni-pozadavky-na-kvalifikovane-eid-systemy-a-eid-prostredky.aspx

NCSC, 2018. *Principle 10: Identity and authentication* [online] [vid. 2022-10-04]. Dostupné z: https://www.ncsc.gov.uk/collection/cloud/the-cloud-security-principles/principle-10-identity-and-authentication

NISO, 2005. *Ranking of Authentication and Access Methods Available to the Metasearch Environment* [online]. 2005. Dostupné z: https://www.niso.org/sites/default/files/2017-08/RP-2005-01.pdf

NIST, 2020. *NIST Special Publication 800-63B* [online] [vid. 2022-10-04]. Dostupné z: https://pages.nist.gov/800-63-3/sp800-63b.html

NOCODE.TECH, 2022. *What is no-code?* [online] [vid. 2022-10-13]. Dostupné z: https://www.nocode.tech/article/what-is-no-code

OAUTH, 2022. *OAuth 2.1* [online] [vid. 2022-11-14]. Dostupné z: https://oauth.net/2.1/

OKTA, 2022a. Example Flow. *OAuth 2.0 Simplified* [online]. [vid. 2022-11-14]. Dostupné z: https://www.oauth.com/oauth2-servers/server-side-apps/example-flow/

OKTA, 2022b. What is Authorization? - Examples and definition. *auth0* [online] [vid. 2022-10-11]. Dostupné z: https://auth0.com/intro-to-iam/what-is-authorization/

ONELOGIN, 2022a. *Biometric Authentication: Good, Bad, & Ugly | OneLogin* [online] [vid. 2022-11-18]. Dostupné z: https://www.onelogin.com/learn/biometric-authentication

ONELOGIN, 2022b. *How Does Single Sign-On (SSO) Work? | OneLogin* [online] [vid. 2022-11-12]. Dostupné z: https://www.google.com/

ONELOGIN, 2022c. *SAML vs OIDC: All You Need to Know | OneLogin* [online] [vid. 2022-11-13]. Dostupné z: https://www.google.ch/

ONELOGIN, 2023. *OTP, TOTP, HOTP: What's the Difference?* | *OneLogin* [online] [vid. 2023-01-15]. Dostupné z: https://www.onelogin.com/learn/how-single-sign-on-works

ORACLE, 2022a. *About LDAP (Understanding the LDAP Binding Component)* [online] [vid. 2022-11-17]. Dostupné z: https://docs.oracle.com/cd/E19182-01/820-6573/6nht2e5a4/index.html

ORACLE, 2022b. *SASL Authentication* [online] [vid. 2022-11-17]. Dostupné z: https://docs.oracle.com/javase/jndi/tutorial/ldap/security/sasl.html

ORACLE, 2022c. X.500 Overview. *X.500 Overview* [online] [vid. 2022-11-17]. Dostupné z: https://docs.oracle.com/javase/jndi/tutorial/ldap/models/x500.html

OUTSYSTEMS, 2022. *What is Low Code | Low Code Guide* [online] [vid. 2022-10-12]. Dostupné z: https://www.outsystems.com/guide/low-code/

OWASP, 2021a. *Multifactor Authentication - OWASP Cheat Sheet Series* [online] [vid. 2022-11-07]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Multifactor_Authentication_Cheat_Sheet.html

OWASP, 2021b. *OWASP Top Ten | OWASP Foundation* [online] [vid. 2022-10-30]. Dostupné z: https://owasp.org/www-project-top-ten/

OWASP, 2022a. *OWASP Top 10 Low-Code/No-Code Security Risks | OWASP Foundation* [online] [vid. 2022-10-25]. Dostupné z: https://owasp.org/www-project-top-10-low-code-no-code-security-risks/

OWASP, 2022b. *Password Storage - OWASP Cheat Sheet Series* [online] [vid. 2022-11-03]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html#salting

OWASP, 2022c. *Session Timeout | OWASP Foundation* [online] [vid. 2022-10-06]. Dostupné z: https://owasp.org/www-community/Session_Timeout

OWASP, 2023. *OWASP Java HTML Sanitizer* [online]. Java. 4. leden 2023. B.m.: OWASP. [vid. 2023-01-06]. Dostupné z: https://github.com/OWASP/java-html-sanitizer

PASSLIB, 2020a. *Modular Crypt Format — Passlib v1.7.4 Documentation* [online] [vid. 2023-01-05]. Dostupné z: https://passlib.readthedocs.io/en/stable/modular_crypt_format.html

PASSLIB, 2020b. *passlib.hash.bcrypt - BCrypt — Passlib v1.7.4 Documentation* [online] [vid. 2023-01-05]. Dostupné z: https://passlib.readthedocs.io/en/stable/lib/passlib.hash.bcrypt.html#passlib.hash.bcrypt

PAVLÍČEK, Luboš, 2020. Identifikace a autentizace. In: . Přednáška. Praha: VŠE.

PEYROTT, Sebastian, 2022. What Is and How Does Single Sign-On Authentication Work? *Auth0 - Blog* [online] [vid. 2022-11-12]. Dostupné z: https://auth0.com/blog/what-is-and-how-does-single-sign-on-work/

PGADMIN, 2023. *pgAdmin - PostgreSQL Tools* [online] [vid. 2023-01-04]. Dostupné z: https://www.pgadmin.org/

RICHARDSON, Clay a John RYMER, 2014. New Development Platforms Emerge For Customer-Facing… *Forrester* [online] [vid. 2022-10-12]. Dostupné z: https://www.forrester.com/report/New-Development-Platforms-Emerge-For-CustomerFacing-Applications/RES113411

RICHER, Justin a Antonio SANSO, 2017. *OAuth 2 in Action*. 1. edice. Shelter Island, NY: Manning Publications Co. ISBN 978-1-61729-327-6.

ROY, Sandip, 2021. *Authentication vs Authorization | Baeldung on Computer Science* [online] [vid. 2022-10-11]. Dostupné z: https://www.baeldung.com/cs/authentication-vs-authorization

SAHAY, Apurvanand, Arsene INDAMUTSA, Davide Di RUSCIO a Alfonso PIERANTONIO, 2020. Supporting the understanding and comparison of low-code development platforms. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* [online]. Portoroz, Slovinsko: IEEE, s. 171–178 [vid. 2022-08-24]. ISBN 978-1-72819-532-2. Dostupné z: doi:10.1109/SEAA51224.2020.00036

SANCHIS, Raquel, Óscar GARCÍA-PERALES, Francisco FRAILE a Raul POLER, 2020. Low-Code as Enabler of Digital Transformation in Manufacturing Industry. *Applied Sciences* [online]. **10**(1), 12. ISSN 2076-3417. Dostupné z: doi:10.3390/app10010012

SAUCE LABS, 2021. Use cases for which low-code platforms are useful worldwide in 2021. *Statista* [online] [vid. 2022-10-14]. Dostupné z: https://www-statista-com.zdroje.vse.cz/statistics/1272131/low-code-platforms-use-cases-worldwide/

SAVVY SECURITY, 2021. 2 Factor Authentication vs 2 Step Verification: What's The Difference? *Savvy Security* [online]. [vid. 2022-10-10]. Dostupné z: https://cheapsslsecurity.com/blog/2-factor-authentication-vs-2-step-verification-whats-the-difference/

SHULER, Kevin, 2022. *(50+) Hyper Relevant Low-Code Statistics And Facts for 2022* [online] [vid. 2022-10-15]. Dostupné z: https://quandarycg.com/low-code-statistics/, https://quandarycg.com/low-code-statistics/

SNYK, 2023. react-native-touch-id vulnerabilities | Snyk. *Find detailed information and remediation guidance for vulnerabilities.* [online] [vid. 2023-02-15]. Dostupné z: https://security.snyk.io/

SPRÁVA ZÁKLADNÍCH REGISTRŮ, 2022. *Identifikační prostředky* [online] [vid. 2022-10-09]. Dostupné z: https://info.identitaobcana.cz/idp/

STRANGE, Warren, 2023. *GoogleAuth* [online]. Java. 2. únor 2023. [vid. 2023-02-07]. Dostupné z: https://github.com/wstrange/GoogleAuth

ŠÍMA, Vít, 2021. *Právní úprava on-line autentizace a identifikace* [online]. Brno [vid. 2022-10-30]. Písemná závěrečná práce. b.n. Dostupné z: https://is.muni.cz/th/f92od/LLM_SIMA_Pravni_uprava_on-line_autentizace_a_identifikace.pdf

TOWNSEND, Alicia, 2021. What is the Real Difference Between SAML and OIDC. *OneLogin Identity Management Blog* [online]. [vid. 2022-11-13]. Dostupné z: https://www.onelogin.com/blog/real-difference-saml-oidc

TRAN, Quang Nhat, 2021. New year, new beginnings with native mobile. *Mendix Community* [online]. [vid. 2023-02-14]. Dostupné z: https://medium.com/mendix/mendix-the-beginning-of-native-mobile-cfdcf5eda969

UML-DIAGRAMS.ORG, 2022. *Unified Modeling Language (UML) description, UML diagram examples, tutorials and reference for all types of UML diagrams - use case diagrams, class, package, component, composite structure diagrams, deployments, activities, interactions, profiles, etc.* [online] [vid. 2022-10-12]. Dostupné z: https://www.uml-diagrams.org/

UNICREDIT BANK, 2019. *Manual-BussinesNet.pdf* [online]. listopad 2019. [vid. 2022-11-19]. Dostupné z: https://www.unicreditbank.cz/content/dam/cee2020-pws-cz/cz-dokumenty/Manual-BussinesNet.pdf

VAN DER HOEK, Jasper, 2018. Mendix Forum - Question Details. In: *Mendix Forum* [online]. [vid. 2023-01-07]. Dostupné z: https://forum.mendix.com/link/space/studio%20pro/questions/86069

VOJINOVIC, Ivana, 2022. Save Your Data with These Empowering Password Statistics. *Dataprot* [online] [vid. 2022-11-01]. Dostupné z: https://dataprot.net/statistics/password-statistics/

WALDRON, Rich, 2022. Council Post: 2022 Predictions: Low Code And The Future Of Work. *Forbes* [online] [vid. 2022-10-14]. Dostupné z: https://www.forbes.com/sites/forbestechcouncil/2022/02/28/2022-predictions-low-code-and-the-future-of-work/

WILSON, Yvonne a Abhishek HINGNIKAR, 2019. *Solving Identity Management in Modern Applications: Demystifying OAuth 2.0, OpenID Connect, and SAML 2.0*. 1. edice. New York, NY: Apress. ISBN 978-1-4842-5095-2.

WONG, Jason, Kimihiko IIJIMA, Adrian LEOW, Akash JAIN a Paul VINCENT, 2021. Magic Quadrant for Enterprise Low-Code Application Platforms. *Gartner* [online] [vid. 2022-08-24]. Dostupné z: https://www.gartner.com/doc/reprints?id=1-27IIPKYV&ct=210923&st=sb

ZAPLETAL, Lukáš, 2000. Lehký úvod do LDAP. *Root.cz* [online] [vid. 2022-11-17]. Dostupné z: https://www.root.cz/clanky/lehky-uvod-do-ldap/

Přílohy

Příloha A: Demo Aplikace

Popis

Pro účely testování a průzkumu autentizačních metod byla vytvořena společně s prací i demo aplikace. Tato aplikace je vytvořena ve verzi Mx 9.18.

Struktura aplikace

Aplikace byla vytvořena z prázdné šablony a používá všechny moduly zmíněné v kapitole 2.1. Dále obsahuje moduly, na kterých jsou tyto moduly dependentní. Celkově jsou použity tyto marketplace moduly:

- Administration
- Atlas_Core
- Atlas_NativeMobile_Content
- Atlas_Web_Content
- CommunityCommons
- DataWidgets
- Email_Connector
- Encryption
- GoogleAuthenticator
- Ldap
- MendixSSO
- MFAmodule
- MxModelReflection
- NanoflowCommons
- NativeMobileResources
- OIDC
- Recaptcha
- WebActions

Implementace samotných metod, stránky a další autorem vytvořené komponenty se nachází ve vlastním modulu Demo. Žádný z modulů, kromě modulu pro LDAP, nebyl modifikován. LDAP modul musel být konvertován a jeho Java akce musely být upraveny, aby jej bylo možné použít.

Role

V aplikaci jsou definovány 3 projektové role – Administrator, User a Guest. Role Administrator je schopná konfigurovat aplikaci, primárně autentizační metody, a spravovat uživatele. Role User je pro uživatele přihlašující se do aplikace a role Guest je anonymní role pro nepřihlášené uživatele, tj. uživatele používající autentizační metody.

Konfigurace aplikace

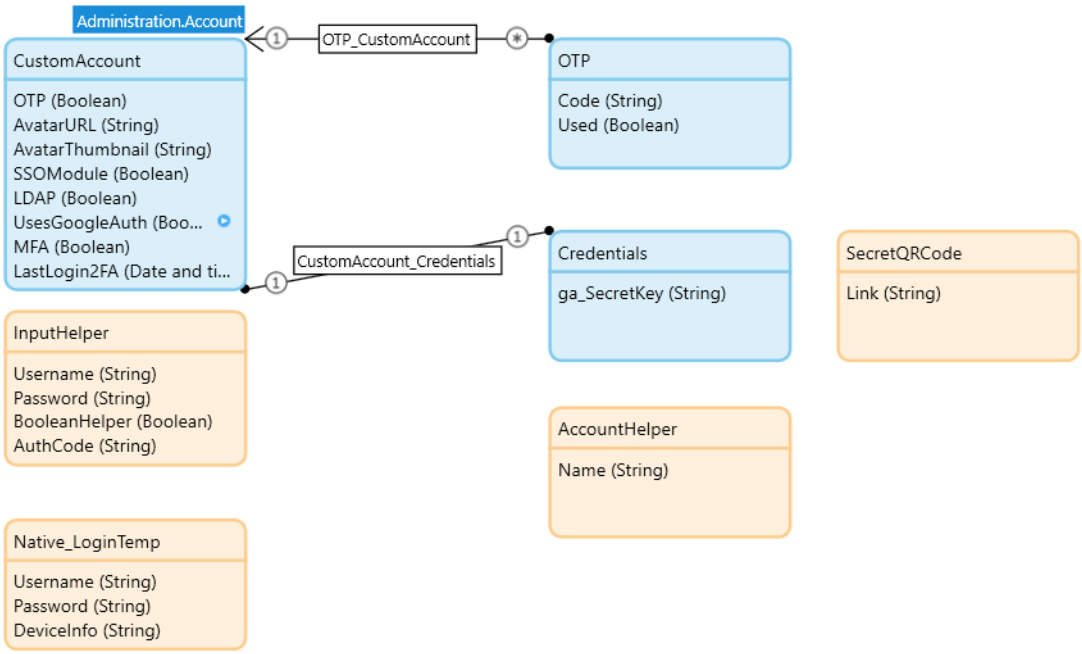
Konfigurace autentizačních metod a modulů se provádí obecně v běžící aplikaci pomocí administrátorského účtu. V systémovém nastavení (např. v runtime) nebyly prováděny žádné změny, kromě nastavení „after startup MF".

Tento MF řídí, spouští a připravuje některé autentizační metody. Je důležité se ujistit, že při spouštění aplikace je povolena maximálně jedna metoda. Při větším počtu může dojít ke komplikacím, kdy se aplikace nemusí spustit či se nebude chovat správně při běhu. Toto se týká MendixSSO, CustomJavaLogin akce, Custom OTP, LDAP a MFAmodule. Všechny tyto komponenty modifikují defaultní login akci.

Vlastní modul

Struktura modulu Demo je následující:

- AfterStartUp MF je možné nalézt v adresáři Lifecycle,
- jednotlivé implementace metod je možné nalézt v adresáři Used, kde pro každou metodu je specifický podadresář,
- login stránka a její custom layout je možné také nalézt v adresáři Used,
- stránky a logiku pro správu uživatelů a administraci obecně je možné nalézt v adresáři Used, podadresář Administration,
- domain model pro autorem vytvořené komponenty a logiku je standardně umístěn v modulu. Zde je vyobrazen na Obrázek 40 níže.



Obrázek 40: Domain model aplikace (zdroj: Autor)

Příloha B: Otázky

Následujících 8 otázek bylo zasláno společně se zkrácenou verzí práce odborníkům z praxe pro validaci:

1. Are all main Mendix authentication methods covered, or did the author forget to include some of them?
2. Do the selected dimensions and criteria cover all important areas of an authentication method? If not, which are missing?
3. Would you agree with the results in chapter 3 Is there anything that should be done/ranked differently or that was overlooked?
4. Could you comment on the results in chapter 4? Do you find these findings interesting and useful? If yes, which?
5. According to the findings and your personal expertise, could you comment on the 5th chapter? Do you agree with recommendations?
6. Could you rate overall quality, level of detail and usefulness of the thesis?
7. Did you find any mistakes or false statements regarding Mendix, authentication methods or any other area?
8. Anything you would like to add? This is an open question for your remaining comments.

Poznámka autora: Čísla kapitol v zasílané, zkrácené verzi jsou jiné, jak čísla kapitol v této diplomové práci. Chapter 3 v otázce odkazuje na kapitolu 6, chapter 4 na kapitolu 7 a chapter 5 na kapitolu 8 této práce.

Příloha C: Odpovědi na Otázky

Odborník A

1. **Are all main Mendix authentication methods covered, or did the author forget to include some of them?**

I guess I'm missing SAML 2.0 via the SAML20 module. But then I guess this is somehow embedded into some of the other methods? Or was it something not possible to test at low/no cost?

2. **Do the selected dimensions and criteria cover all important areas of an authentication method? If not, which are missing?**

I think it's covered, but as I will explain in #3, I don't think the weight is correct, and there might be a decently big hole in the real-life application of the testing method.

3. **Would you agree with the results in chapter 3 Is there anything that should be done/ranked differently or that was overlooked?**

Poznámka autora: Pojmem „COMPANY" je nahrazen název odborníkovy společnosti. Dále „table 3.1.2 on page 49" odkazuje na Tabulku 13.

I'm quite surprised username/password scored so high, and especially so much so that it scored higher than Mendix SSO and the same as OIDC SSO and LDAP in usability. Because at least for COMPANY, SSO is just one click. It should score higher than username/password, even if using a password manager (and if so, then you have another app, which means it should be "yes" in table 3.1.2 on page 49). If you don't use a password manager – and most users don't – then the only way to be fast is by reusing passwords, which is not secure at all. Else, you have passwords written somewhere else, in which case searching for the correct password would then take longer than it would with SSO, even with MFA factored in.

Just for reference: within COMPANY, I am a member of about 70+ environments in Mendix. This does not include all of the other company apps that require username + password authentication that I also need to use. And on top of all this, I also have all of my private apps and services that I need username + password for. If I don't use SSO, and I don't use a password manager, it will take me way longer than 10 seconds just looking for the credentials I need.

And since the whole point of authentication is security – not ease of use – the security factors should have a much higher weight than usability.

4. **Could you comment on the results in chapter 4? Do you find these findings interesting and useful? If yes, which?**

Same comment as above, I'm quite surprised that username + password scored so high. Which, to be honest, leads me to question the method. Because from my experience, both as a user and an administrator, this is not better than SSO, or even OTP.

5. **According to the findings and your personal expertise, could you comment on the 5th chapter? Do you agree with recommendations?**

Poznámka autora – 5.1 pojednává o firemním ERP, 5.2 o e-shopu a 5.3. o zpravodajském webu.

I can only speak for section 5.1 since I do not have experience in 5.2 and 5.3. I agree with the recommendations, but not certain statements made, such as:

"The applications do not have to work with any anonymous users." Not always true. Some apps require anonymity.

"Users may be uncomfortable using SSO or multi-factor authentication…" as I mentioned above, when the alternative is managing hundreds of credentials, SSO, MFA, or OTP is a far more convenient method than username + password.

6. **Could you rate overall quality, level of detail and usefulness of the thesis?**

I would say the detail is quite high. The only criticisms I can think of, as mentioned, are:

1. The weights given to the criteria, and

2. That the test method seems to be limited to the context of a single app, which is not realistic. Even for my own private purposes, I have tens of places I log in to with username + password, that I wish there was some easier way of doing so. On a mobile device, biometrics (finger scan or face ID) makes it easier, but on a computer, not quite.

7. **Did you find any mistakes or false statements regarding Mendix, authentication methods or any other area?**

I did not find any factually false statements in the paper.

8. **Anything you would like to add? This is an open question for your remaining comments.**

Good work overall.

Odborník B

1. **Are all main Mendix authentication methods covered, or did the author forget to include some of them?**

Looks good to me.

2. **Do the selected dimensions and criteria cover all important areas of an authentication method? If not, which are missing?**

Poznámka autora: Pojmem „COMPANY" je nahrazen název odborníkovy společnosti.

Missing topics:

Compliance (GDPR ok?), Integration with other systems, flexibility -> How can specific Method be adjusted according to the needs of a enterprise company like COMPANY.

Dependencies from Third-party modules (CVE's come for SAML every month!).

3. **Would you agree with the results in chapter 3 Is there anything that should be done/ranked differently or that was overlooked?**

Our Cybersec does not recommend to use password / user. Instead token-based authentication is recommended.

4. **Could you comment on the results in chapter 4? Do you find these findings interesting and useful? If yes, which?**

User+Password are generally not seen as very secure, since user use poor passwords and often not password manager. I would enlight this aspect in the thesis. Technically passwords are easy to implement, but from a security pov it's a highly critical topic.

5. **According to the findings and your personal expertise, could you comment on the 5th chapter? Do you agree with recommendations?**

E-Shops and News-Website are no typical use-cases for Mendix apps. Mendix itself has a strong focus for replacing legacy company apps, that are mostly used internally (former excel sheets, VBA, MS Access…) Put more focus on this aspect.

6. **Could you rate overall quality, level of detail and usefulness of the thesis?**

Poznámka autora: neobdržena odpověď.

7. **Did you find any mistakes or false statements regarding Mendix, authentication methods or any other area?**

Poznámka autora: neobdržena odpověď.

8. **Anything you would like to add? This is an open question for your remaining comments.**

Poznámka autora: neobdržena odpověď.

Odborník C

1. **Are all main Mendix authentication methods covered, or did the author forget to include some of them?**

Poznámka autora: Pojmem „COMPANY" je nahrazen název odborníkovy společnosti.

Yes. At COMPANY we primarily use SSO via SAML or OIDC and manual login with username and password. OTP or MFA we not really have in apps, but SaaS components of Mendix itself.

2. **Do the selected dimensions and criteria cover all important areas of an authentication method? If not, which are missing?**

From my perspective the dimensions suffice.

3. **Would you agree with the results in chapter 3 Is there anything that should be done/ranked differently or that was overlooked?**

Looks good to me.

4. **Could you comment on the results in chapter 4? Do you find these findings interesting and useful? If yes, which?**

Poznámka autora: Pojmem „COMPANY" je nahrazen název odborníkovy společnosti.

What is surprising is that username and password is considered as best balanced alongside all dimensions. Given the many COMPANY Security Advisory related to those modules, but also considering the human factor. To be fair, there is also a lot of SAML related COMPANY Security Advisories.

5. **According to the findings and your personal expertise, could you comment on the 5th chapter? Do you agree with recommendations?**

In general yes. Two things to be considered from my personal experience. What is challenging for us is managing users across all of our apps. A central user repository works best for us. If a user leaves the company, access is blocked in all apps. At least from a SSO perspective. Local users are out of reach. The fact that working with username and password cannot be securely prevented is quite a concern. You have to trust your developers and local admins a lot.

6. **Could you rate overall quality, level of detail and usefulness of the thesis?**

Very useful to me and us. If ever you or a colleague of yours going to write another paper like this, what about focusing on in-app encryption options and storing secrets (e.g. constants? Db?)?

7. **Did you find any mistakes or false statements regarding Mendix, authentication methods or any other area?**

No.

8. **Anything you would like to add? This is an open question for your remaining comments.**

Thanks for sharing your work with us.

Odborník D

1. **Are all main Mendix authentication methods covered, or did the author forget to include some of them?**

As far as I know, this covers all of the possible methods of authentication in Mendix.

2. **Do the selected dimensions and criteria cover all important areas of an authentication method? If not, which are missing?**

Usability dimension does not take into account the benefits of centralized user management. The criteria should also cover situations where one login would allow user to use multiple applications without additional logins.

3. **Would you agree with the results in chapter 3 Is there anything that should be done/ranked differently or that was overlooked?**

The user comfort could be assessed differently to take into account my point from the previous question. The metric does not factor in the real life situation where a user forgets his/hers password(s).

4. **Could you comment on the results in chapter 4? Do you find these findings interesting and useful? If yes, which?**

In my experience SSO method tends to be more user friendly and secure than the "good old" name and password approach. It provides more detailed overviews and information for administrators, can be changed when the application is running as well as enabling easier user management.

5. **According to the findings and your personal expertise, could you comment on the 5th chapter? Do you agree with recommendations?**

From a perspective of a developer who has worked on multiple corporate applications the recommendations for corporate ERP are reasonable. As for the other two applications,

although not having any experience working with these, the recommended methods seem adequate.

6. **Could you rate overall quality, level of detail and usefulness of the thesis?**

The thesis is quite detailed. I appreciate that the author has had to dive into undocumented java territory and provided insights on the inner workings of given methods. I can see these insights being used in the future when developing/customizing authentication modules.

7. **Did you find any mistakes or false statements regarding Mendix, authentication methods or any other area?**

No.

8. **Anything you would like to add? This is an open question for your remaining comments.**

Nothing.