FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

| | |
|---|---|
| **Název:** | Linky - jádro systému |
| **Student:** | Libor Plíšek |
| **Vedoucí:** | Ing. Jiří Chludil |
| **Studijní program:** | Informatika |
| **Studijní obor:** | Webové a softwarové inženýrství |
| **Katedra:** | Katedra softwarového inženýrství |
| **Platnost zadání:** | Do konce zimního semestru 2019/20 |

## Pokyny pro vypracování

Linky je interaktivní světelná instalace na jižní straně fasády FEL ČVUT v Praze. Cílem práce je návrh a implementace jádra, které bude umožňovat ovládání celého systému, funkcionalitu nainstalovaných programů, klientské Android aplikace a zprostředkovávat komunikaci mezi nimi.
1. Analyzujte možnosti stávajícího systému Linky.
2. Navrhněte funkce pro přihlášení/registraci pomocí Google a nebo Facebook účtu.
3. Navrhněte způsob spouštění programů podle administrátorem nastaveného plánu.
4. Navrhněte způsob, jak přidávat, odebírat, spouštět a ukončovat programy.
5. Navrhněte komunikaci mezi jádrem systému a programy, ovládací aplikací, administračním rozhraním a veřejnými komponentami.
6. Navržené řešení implementujte pomocí technologie NodeJS. Pro nasazení použijte CI a verzovací systém GIT.
7. Hotový systém podrobte vhodným testům.

## Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 11. dubna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

# Linky – jádro systému

***Libor Plíšek***

Katedra softwarového inženýrství
Vedoucí práce: Ing. Jiří Chludil

13. května 2019

# Poděkování

Prvně děkuji svému vedoucímu Ing. Jiřímu Chludilovi za ukázkový přístup při tvorbě bakalářské práce, za veškeré rady, a hlavně za jeho čas, který mi vždy ochotně věnoval. Neméně děkuji své přítelkyni Veronice Straškové za znamenitou podporu během celého studia. A v neposlední řadě děkuji svým rodičům za to, že mi vůbec umožnili studium na této prestižní škole.

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 13. května 2019 ……

### Odkaz na tuto práci

Plíšek, Libor. *Linky – jádro systému*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2019.

# Abstrakt

Projekt Linky je interaktivní světelná instalace na jižní straně fasády budovy FEL ČVUT v Praze. Tato bakalářská práce se zabývá tvorbou softwarového jádra, které umožňuje správný chod světelné fasády. Jádro poskytuje ovládání celého informačního systému, spravuje a řídí nainstalované serverové aplikace, obsluhuje klientské aplikace a zprostředkovává komunikaci mezi nimi.

Obsahem této bakalářské práce je analýza existujících řešení a podrobná analýza stávajícího a nového informačního systému. Další součástí této práce je detailní návrh nového informačního systému. Po kompletním návrhu je v této práci popsán postup implementace pomocí technologií Node.js a Express.js za použití průběžné integrace (continuous integration) a verzovacího systému Git. Jako poslední je zde popsán průběh testování a možný budoucí vývoj softwarového jádra i celého informačního systému.

**Klíčová slova** Linky FEL ČVUT, serverová aplikace, backend, jádro, světelná instalace, světelná fasáda, REST, API, informační systém, Node.js

# Abstract

The Linky project is an interactive light installation on the south side of the facade of the CTU building in Prague. This bachelor thesis deals with the creation of software core, which enables correct operation of the light facade. The core provides control of the entire information system, manages and control installed server applications, operates client applications, and mediate communication between them.

The content of this thesis is an analysis of existing solutions and a detailed analysis of the current and new information system. Another part of this work is a detailed design of a new information system. After the complete design, this work describes the implementation process using Node.js and Express.js technologies using continuous integration and the Git versioning system. Lastly, there are described the progress of testing and possible future development of the software core and the whole information system.

**Keywords** Linky FEL CTU, server application, backend, core, light installation, light facade, REST, API, information system, Node.js

# Obsah

# Seznam obrázků

# Seznam tabulek

# Úvod

Již těžko si lze představit, že by dnešní moderní svět existoval bez informačních systémů. Stejně tak se i tato bakalářská práce zabývá tvorbou softwarového jádra nového informačního systému, který zajišťuje správný běh světelné instalace Linky na jižní straně fasády budovy FEL ČVUT v Praze.

Použití světelných fasád a velkoplošných obrazovek je v posledním desetiletí v našem světě velmi populární. Většina světelných instalací, především velkoplošné obrazovky, se používají hlavně pro reklamní a informační účely. Linky jsou jiné, kromě informačních účelů je to především dekorativní prvek, který má za úkol oživovat prostor dejvického kampusu nejen pro studenty a zaměstnance ČVUT, ale i pro širokou veřejnost. Tato světelná fasáda nabízí širokou škálu barevných vzorců, dokáže zobrazit jakékoliv vizualizace s nejrůznějšími druhy podnětů.

Dlouho mě lákalo podílet se na vývoji projektu, který je přístupný široké veřejnosti, který je vidět, kolem kterého každý den projde mnoho lidí, právě proto jsem si vybral toto téma bakalářské práce. Stávající informační systém projektu Linky není uspokojivě navržený, není ani kompletní a jsou s ním bezpečnostní a výkonnostní problémy, proto bylo nezbytné vytvoření nového informačního systému, který odstraní všechny jeho nedostatky.

V první kapitole této bakalářské práce se čtenář může seznámit s cílem teoretické a praktické části práce. V kapitole 2 je možné se dočíst podrobnosti o analýze existujících řešení a analýze stávajícího i nového informačního systému. Ve 3. kapitole je popsán detailní návrh softwarového jádra i celého informačního systému. V kapitole 4 je zpracován průběh realizace za pomocí technologií Node.js a Express.js, průběžné integrace (continuous integration) a verzovacího systému Git. A nakonec v páté kapitole je popsán průběh testování realizovaného řešení.

Kapitola 1

# Cíl práce

Hlavním cílem této práce je tvorba softwarového jádra, které umožňuje ovládání celého nového informačního systému. Softwarové jádro má za cíl poskytovat přihlašování a správu uživatelů, správu a řízení serverových aplikací, správu a obsluhu klientských aplikací a zprostředkovávat komunikaci mezi nimi.

Cílem teoretické části práce je analýza existujících řešení a analýza stávajícího i nového informačního systému. Dále návrh přihlašování a registrace uživatelů pomocí externích Facebook a Google účtů, návrh způsobu, jak přidávat, odebírat, spouštět a ukončovat serverové aplikace, včetně automatického spouštění aplikací podle předem nastaveného plánu. A v neposlední řadě komplexní návrh veškeré komunikace mezi jádrem informačního systému, serverovými aplikacemi, klientskými aplikacemi, webovým portálem, grafickou knihovnou a veřejnými komponentami.

Cílem praktické části práce je implementace všech navržených řešení z teoretické části této práce za pomocí technologie Node.js, průběžné integrace (continuous integration) a verzovacího systému Git. Následně pak hotové řešení řádně otestovat pomocí vhodných testů a vytvořit vhodnou dokumentaci.

Kapitola 2

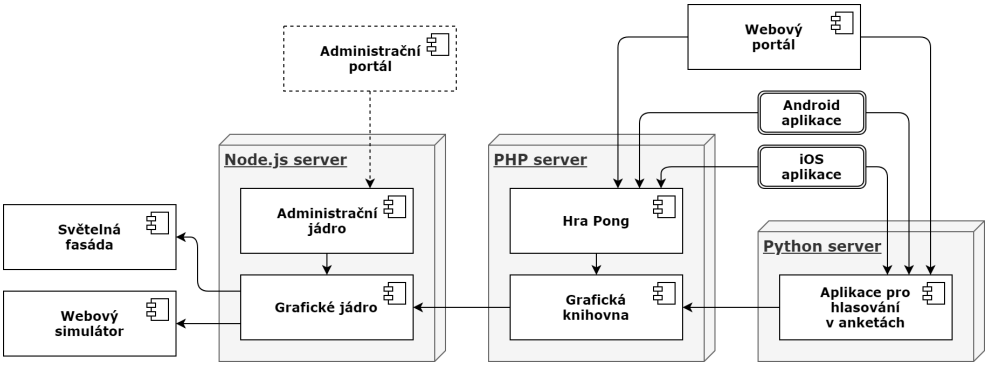
# Analýza

V této kapitole se nachází analýza existujících řešení, analýza stávajícího řešení (i jeho aktualizace) a nakonec analýza nového řešení, včetně požadavků.

## 2.1 Existující řešení

Venkovní světelné instalace, velkoplošné obrazovky a projekce na fasády budov existují po celém světě. Světelná instalace Linky na fasádě budovy FEL ČVUT je i přes to jedinečná. Unikátnost této světelné instalaci zajišťuje seskupení přes 1000 světelných prvků do pěti samostatných sloupců. Tyto sloupce jsou od sebe stejně vzdálené a jako celek tvoří přibližně čtverec.

Obrázek 2.1: Fotografie světelné instalace Linky na budově FEL ČVUT [1]




Obrázek 2.2: Schéma stávajícího informačního systému [4]

## 2.2 Stávající řešení

Tato podkapitola obsahuje analýzu stávajícího informačního systému (IS), který aktuálně obsluhuje světelnou fasádu. Část této analýzy vychází z týmového projektu řešeného v rámci předmětů BI-SP1 [2] a BI-SP2 [3] ve 2. ročníku, kde jsem se tímto tématem zabýval spolu s ostatními členy svého týmu.

Na obrázku 2.2 je znázorněno schéma stávajícího informačního systému, který se skládá ze čtyř základních částí, které jsou v této analýze následně zvlášť rozebrány.

**Seznam základních částí systému**

- Node.js server
- PHP server
- Python server
- Mobilní aplikace

### Node.js server

Node.js server [5] je jádrem informačního systému. Hostuje jednoduché grafické jádro, pomocí kterého lze měnit intenzitu a barvy jednotlivých prvků světelné instalace. Toto jádro přijímá požadavky pomocí REST API [6] a k ověřování požadavků se používají API klíče.

Správu API klíčů by bylo možné provádět pomocí administračního portálu, který bohužel není kompletně dokončen. Ve chvíli, kdy jsme prováděli týmovou analýzu, byl vystavený pouze jeden API klíč, který byl veřejně dostupný v online dokumentaci, takže světelnou fasádu mohl ovládat prakticky kdokoliv.

Výstup z Node.js serveru lze přesměrovat kromě světelné fasády, také do webového simulátoru. Tento simulátor slouží k testování informačního systému po úpravách serverových aplikací, nebo před nasazením úplně nových.

### PHP server

PHP server [7] obsahuje kompletní implementaci hry Pong [8]. S tímto serverem komunikují mobilní aplikace pro platformy Android [9] a iOS [10] a mimo jiné i webový portál. Server zároveň obsahuje komplexní grafickou knihovnu a umožňuje vývojářům (a jejich serverovým aplikacím) komunikovat s Node.js serverem o něco jednodušeji. Tuto knihovnu využívají i jiné serverové aplikace, například aplikace pro hlasování v anketách.

### Python server

Python server [11] hostuje serverovou aplikaci pro hlasování v anketách. Tato aplikace komunikuje s webovým portálem, přes který lze vytvořit anketní otázky a dále komunikuje i s mobilními zařízeními, přes která mohou uživatelé hlasovat. Výsledky hlasování se pak promítají na světelné fasádě.

### Mobilní aplikace

Mobilní aplikace podporuje velmi jednoduché přihlášení uživatele (pouze na základě zadání přihlašovacího jména), podporuje ovládání hry Pong, a také aplikaci pro hlasování v anketách. Při ovládání hry Pong stávající informační systém nestíhá zpracovávat požadavky a hra není plynulá.

### Shrnutí

Aktuální řešení informačního systému poskytuje dobrý základ, kterým je především Node.js server, jehož REST API se dá v novém informačním systému pohodlně využít. Dále se mohou po úpravách zachovat i již naprogramované serverové aplikace jako je hra Pong, nebo aplikace pro hlasování v anketách. Ostatní komponenty by bylo dobré nahradit zcela novými.

#### Klady

- jednoduché REST API Node.js serveru
- možnost přesměrování výstupu do webového simulátoru
- ověřování požadavků pomocí API klíčů

**Zápory**

- nedokončená správa API klíčů
- interaktivní serverové aplikace nelze přidávat a upravovat jednoduše (bez aktualizací jádra systému a mobilních aplikací)
- serverové aplikace nelze spouštět automaticky podle předem nastaveného plánu (nastaveným např. správcem systému)
- není navržen žádný koncept, jak se mobilní aplikace mohou připojovat k serverovým aplikacím
- přihlašování uživatelů je zcela nevyhovující (pouze na základě přihlašovacího jména)
- výkon systému je nedostačující (nebo nesprávně optimalizovaný)
- dokumentace systému je zcela nedostatečná

## 2.3 Aktualizované stávající řešení

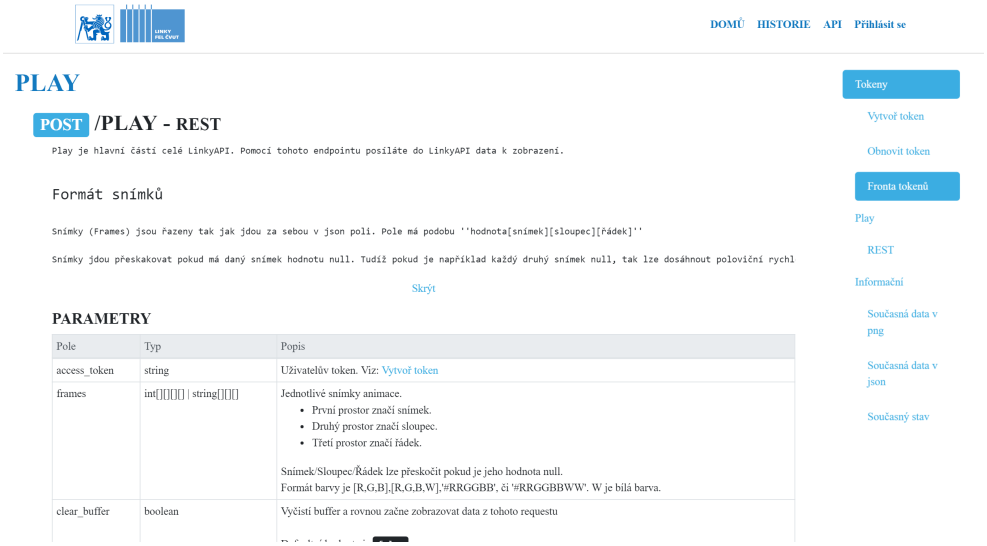
Na začátku roku 2019 bylo spuštěno nové REST API na Node.js serveru, díky kterému lze obsluhovat světelnou fasádu komplexněji [12]. Mezi hlavní změny se řadí podpora snímků. Serverové aplikace již neodesílají informace o jednotlivých bodech obrazu, nýbrž informace o celých obrazových snímcích. Jednotlivé snímky lze přidávat i do vyrovnávací paměti, což zajišťuje především plynulost přechodů a animací.

Nové rozhraní informačního systému je zcela odlišné od původního řešení, takže existující aplikace jsou nekompatibilní a je třeba jej přeprogramovat. Spolu s aktualizovaným řešením byly vytvořeny nové serverové aplikace, takže chod světelné fasády není touto aktualizací přímo omezen.

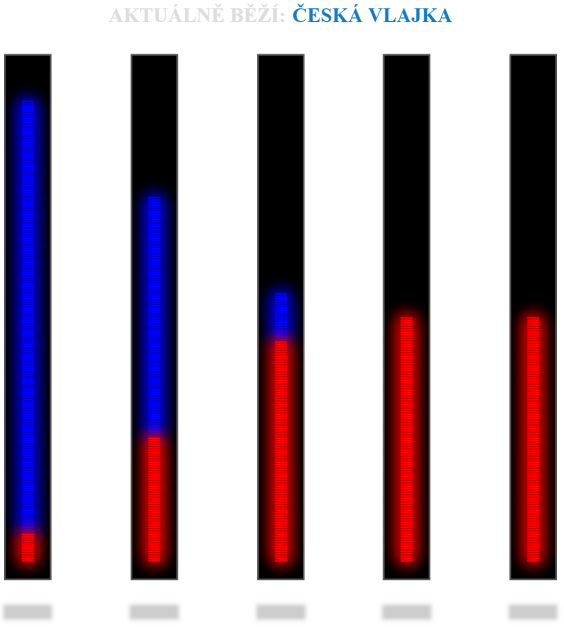
Aktualizované řešení již obsahuje plně funkční podporu API klíčů, čímž se zvýšila bezpečnost IS, a také byla vytvořena nová, lépe formátovaná, online dokumentace [13]. Tato nová dokumentace se naneštěstí neshoduje s reálnou implementací, čímž se stává opět těžko použitelnou.

Součástí aktualizovaného řešení je i nový webový portál, kde mají vývojáři možnost vyzkoušet si nové rozhraní. Testování je prováděno v odděleném prostředí a výstup je směrován na webovou stránku se simulátorem.

Mobilní zařízení stále nelze připojovat automaticky a nelze jej serverovými aplikacemi jednotně obsluhovat. V tomto řešení by každá interaktivní serverová aplikace potřebovala vlastní mobilní aplikaci, a to je jak pro vývojáře serverových aplikací, tak i pro uživatelé systému naprosto nepraktické.




Obrázek 2.3: Ukázka API dokumentace aktualizovaného systému




Obrázek 2.4: Ukázka simulátoru aktualizovaného systému

### Shrnutí

Aktualizované řešení informačního systému poskytuje o něco lepší základ než původní řešení. Největší nevýhody a nedostatky ovšem stále zůstávají zcela nevyřešené.

#### Klady

- komplexní REST API Node.js serveru
- možnost přesměrování výstupu do webového simulátoru
- správa a ověřování požadavků pomocí API klíčů
- vylepšení výkonu (pomocí vykreslování celých snímků)

#### Zápory

- dokumentace systému je nevyhovující
- interaktivní serverové aplikace nelze přidávat a upravovat jednoduše (bez aktualizací jádra systému a mobilních aplikací)
- není navržen žádný koncept, jak se mobilní aplikace mohou připojovat k serverovým aplikacím
- není navržen žádný koncept, jak se uživatelé systému mohou přihlašovat do mobilních aplikací

## 2.4 Nové řešení

Aby bylo přidávání a úprava serverových aplikací jednodušší a nebylo nutné vydávat aktualizace jádra systému a mobilních aplikací, je třeba zavést komunikační konvence.

Po zavedení stejného způsobu komunikace, mezi jádrem systému a serverovými aplikacemi, budou všechny aplikace stejným způsobem obsluhovány a bude se dát mezi nimi jednoduše přepínat. Díky tomu bude jednoduché spouštět aplikace automaticky podle předem nastaveného plánu nebo i samotnými uživateli informačního systému.

Dále je žádoucí vytvořit komunikační konvenci mezi interaktivními serverovými aplikacemi a mobilními zařízeními, takže po úpravě serverové aplikace již nebude nutné vydávat aktualizace mobilních aplikací. Pro jednotné ovládání interaktivních serverových aplikací se musí zavést především konvence v rozvržení ovládacích prvků na displejích mobilních zařízení. Serverové aplikace pak musí na začátku svého běhu poskytnout všem obsluhovaným mobilním zařízením informaci o rozvržení těchto ovládacích prvků.

Do mobilních aplikací je třeba přidat nové přihlašování uživatelů, nejlépe pomocí externích účtů jako je například *Facebook účet* [14] nebo *Google účet* [15], aby se uživatelé systému nemuseli zdržovat vyplňováním údajů a také aby bylo přihlašování bezpečnější. Je třeba navrhnout a vytvořit více uživatelských oprávnění jako jsou například správce systému, nebo vývojář.

Každá serverová aplikace se pak bude lišit svou složitostí. Automatické animační a vizualizační aplikace budou jednodušší, protože se nebudou muset starat o obsluhu mobilních zařízení.

Aby se zjednodušilo i samotné programování interaktivních serverových aplikací, jádro systému bude obsahovat jednotnou frontu uživatelů, kteří se budou chtít připojit k serverové aplikaci přes svoje mobilní zařízení. Při spouštění serverové aplikace pak jádro systému vybere z fronty daný počet uživatelů a serverové aplikaci je automaticky přiřadí. Aplikace tedy nemusí obsahovat žádný složitý mechanismus vybírání uživatelů a na každého uživatele se postupně vždy časem dostane.

### Funkční požadavky

Tato sekce obsahuje analýzu funkčních požadavků jádra nového informačního systému. Následující tabulka 2.1 shrnuje seznam všech funkčních požadavků. U každého požadavku je uveden identifikátor (ID), název a priorita. Na tyto funkční požadavky následně přímo navazují případy užití navržené v následující kapitole.

| **ID** | **Název** | **Priorita** |
|---|---|---|
| FR1 | Základní funkce | střední |
| FR2 | Přihlašování uživatelů | vysoká |
| FR3 | Správa uživatelů | vysoká |
| FR4 | Správa mobilních zařízení | nízká |
| FR5 | Správa serverových aplikací | vysoká |
| FR6 | Ovládání serverových aplikací | vysoká |
| FR7 | Správa prostředí | střední |
| FR8 | Plán běhu serverových aplikací | vysoká |
| FR9 | Fronta mobilních zařízení | střední |

Tabulka 2.1: Seznam funkčních požadavků jádra nového systému

#### FR1 – Základní funkce

Jádro systému poskytuje základní informace o svém stavu a o stavu celého informačního systému. Poskytuje například informace o právě spuštěné serverové aplikaci, nebo o plánu běhu příštích serverových aplikací.

### FR2 – Přihlašování uživatelů

Uživatelé mají možnost přihlásit se do systému a každému je přiřazena uživatelská role. Přihlášení uživatelé mají možnost interagovat se světelnou fasádou pomocí svých mobilních zařízení a pomocí nainstalovaných serverových aplikací. Vývojáři mohou vytvářet a přidávat do systému nové serverové aplikace. A správci systému mohou řídit celý informační systém.

### FR3 – Správa uživatelů

Každý uživatel má možnost spravovat svůj uživatelský účet, tj. zobrazit, editovat, nebo smazat svá osobní evidovaná data.

Správce systému má možnost spravovat účet libovolného uživatele, libovolně zobrazovat, editovat a odstraňovat všechna důležitá evidovaná data. Dále má možnost libovolného uživatele zablokovat, a to na dobu neurčitou (například za porušení pravidel nového systému).

### FR4 – Správa mobilních zařízení

Mobilní zařízení se po připojení do systému evidují automaticky a mohou aktualizovat svá evidovaná data i bez zásahu uživatele. Například aktualizovat verzi nainstalované mobilní aplikace, nebo různé komunikační tokeny.

Správce systému může libovolné zařízení spravovat, nebo zablokovat a znemožnit tak přístup všem uživatelům, kteří dané zařízení používají.

### FR5 – Správa serverových aplikací

Vývojáři mohou přidat do systému nové serverové aplikace, tyto aplikace pak před samotným zveřejněním musí projít testováním a následným schválením správcem systému. Vývojáři mohou libovolně spravovat své vlastní serverové aplikace a testovat je v testovacím prostředí.

Správce systému může kompletně spravovat evidovaná data o libovolné serverové aplikaci.

### FR6 – Ovládání serverových aplikací

Systém umožňuje správcům spouštět a ukončovat serverové aplikace, a to jak okamžitě, tak i automaticky podle předem nastaveného plánu. Předem nastavený plán běhu serverových aplikací obsluhuje systém automatizovaně bez přímého zásahu uživatelů.

### FR7 – Správa prostředí

Správce systému může spravovat prostředí. V systému je možné vytvořit jedno produkční prostředí a zároveň i několik testovacích prostředí pro vývoj a testování serverových aplikací.

Testovací prostředí zajišťuje naprosto stejné podmínky jako produkční, pouze s jediným rozdílem. Obrazová data serverové aplikace se v testovacím prostředí neodesílají na světelnou fasádu, nýbrž do webového simulátoru. K tomuto simulátoru mají přístup libovolní vývojáři a správci systému.

#### FR8 – Plán běhu serverových aplikací

Jádro systému obsluhuje plán běhu serverových aplikací pro každé prostředí zvlášť. Správce systému může tento plán libovolně spravovat a modifikovat, tj. přidávat nové serverové aplikace, editovat jejich evidovaná data, nebo je i odebírat. Dále jádro eviduje kompletní historii běhu všech serverových aplikací, včetně nejrůznějších detailů.

#### FR9 – Fronta mobilních zařízení

Jádro systému udržuje frontu uživatelů pro každé prostředí a automaticky přiřazuje uživatelé z fronty k serverovým aplikacím. Každý přihlášený uživatel má možnost se do fronty přidat a následně se pak připojit k serverové aplikaci.

### Nefunkční požadavky

Tato sekce obsahuje analýzu nefunkčních požadavků jádra nového informačního systému. Následující tabulka 2.2 shrnuje kompletní seznam všech nefunkčních požadavků. U každého požadavku se nachází identifikátor (ID), název a priorita.

| ID | Název | Priorita |
|---|---|---|
| NR1 | Výkon a optimalizace | střední |
| NR2 | Dostupnost a zálohování | vysoká |
| NR3 | Rozšiřitelnost | vysoká |
| NR4 | Bezpečnost | vysoká |

Tabulka 2.2: Seznam nefunkčních požadavků jádra nového systému

#### NR1 – Výkon a optimalizace

Výkon jádra systému by měl být vysoký, ale není to vysokou prioritou. Jádro informačního systému nijak nepřistupuje přímo na světelnou fasádu, takže jeho výkon nikdy přímo neomezuje rychlost streamování a vykreslování dat. I přes to je jádro systému optimalizované a každý požadavek je zpracován poměrně v krátké době.

### NR2 – Dostupnost a zálohování

Informační systém je dostupný přesně tak, jak to umožňuje server, na kterém je produkční řešení jádra systému zprovozněno. Zdrojový kód jádra systému je automaticky zálohovaný, jelikož pro jeho nasazení je použit systém Git [16] přes fakultní online GitLab repozitář [17]. Produkční databáze jádra systému je pravidelně zálohovaná programem běžícím na pozadí serveru.

### NR3 – Rozšiřitelnost

Celé jádro systému je řádně dokumentováno a zcela připraveno pro libovolně rozsáhlé rozšíření. Ve verzovacím systému Git je nastavená průběžná integrace (continuous integration) [18] a tím pádem každá nová změna jádra systému se automaticky ihned projeví přímo na serveru v aktuálním produkčním řešení. Před nasazením nové verze řešení je jádro systému automaticky testováno a při nasazování nové verze je vždy na malý okamžik nedostupné. Pokud při testování dojde k neúspěchu, je vývojář jádra automaticky varován prostřednictvím e-mailu a k nasazení nového řešení nedojde.

Jádro systému je vytvořené v technologii Node.js, ale REST API jádra je v obecně vysoce rozšířeném formátu, takže je přístupné pro libovolnou platformu nebo klientskou aplikaci.

### NR4 – Bezpečnost

V celém REST API jádra systému se používá bezpečný protokol HTTPS [19] a autentizační tokeny. Dále jakýkoliv uživatel, mobilní zařízení nebo i serverová aplikace se dají zablokovat správcem systému, pokud nedodržují pravidla informačního systému. Systém tak splňuje všechny základní bezpečnostní požadavky a opatření.

Kapitola 3

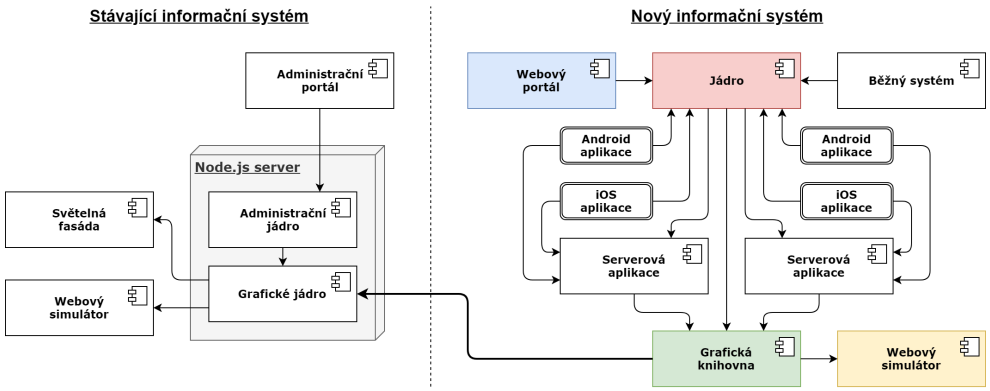
# Návrh

V této kapitole se nachází návrh jádra nového informačního systému. Nejprve je zde popsán nový informační systém, poté jsou zde definovány druhy systémů a uživatelské role. Následně je v této kapitole popsán návrh případů užití a návrh databáze. Detailní návrh případů užití je obsáhlý, a proto se nachází až na konci této práce v příloze C. Jako poslední se na konci této kapitoly se nachází návrh API a návrh démonů.

## 3.1 Nový informační systém

Aby nebyl nový informační systém vázán na jednu konkrétní technologii a byl ve všech ohledech jednoduše rozšiřitelný, skládá se z mnoha jednotlivých komponent, které dohromady tvoří ucelené komplexní řešení.

Na obrázku 3.1 se nachází schéma nového informačního systému napojeného na *Node.js server* ze stávajícího informačního systému (důvod tohoto napojení je popsán v předchozí kapitole).




Obrázek 3.1: Schéma nového informačního systému

Tato bakalářská práce je zaměřena především na vývoj jádra nového informačního systému, všechny ostatní komponenty mají své vlastní návrháře a vývojáře. Návrh nového informačního systému vznikl v týmové spolupráci s vývojáři webového portálu a grafické knihovny. Student *Tomáš Vošický* je hlavním vývojářem webového portálu označeným na obrázku 3.1 modrou barvou a student *Jiří Košata* je vývojářem grafické knihovny označené barvou zelenou. Oba zmínění studenti spolupracují ještě na vývoji webového simulátoru a všechny komponenty vyvíjí v rámci svých bakalářských prací.

## 3.2 Druhy systémů

Ještě před samotným modelováním případů užití je zde uveden přehled druhů systémů nového informačního systému, se kterými, konkrétními způsoby, jádro systému komunikuje.

### Běžný systém

Externí weby a jiný software třetích stran, který komunikuje s jádrem systému, ale není nedílnou součástí nového informačního systému, je v této práci nazýván běžným systémem. Tyto weby a aplikace mohou přistupovat pouze na minimum API endpointů, pouze na takové, kde se nevyžaduje žádná autentizace. Na takových endpointech lze zjistit pouze základní informace, například aktuální stav informačního systému nebo stav jádra.

### Mobilní zařízení

Systémem mobilní zařízení se označuje mobilní aplikace Linky pro platformy Android a iOS. Obě tyto aplikace mohou přistupovat na předem určené API endpointy jádra systému a na API endpointy serverových aplikací. Přes mobilní zařízení mohou přihlášení uživatelé využívat interaktivity informačního systému, mohou například pomocí svého mobilního zařízení vytvářet animace nebo hrát hry, a to přímo na světelné fasádě. Fantazii zde jádro systému meze naklade. To, co mohou uživatelé na fasádě vykreslovat, záleží jen na jejich kreativitě a na možnostech serverových aplikací.

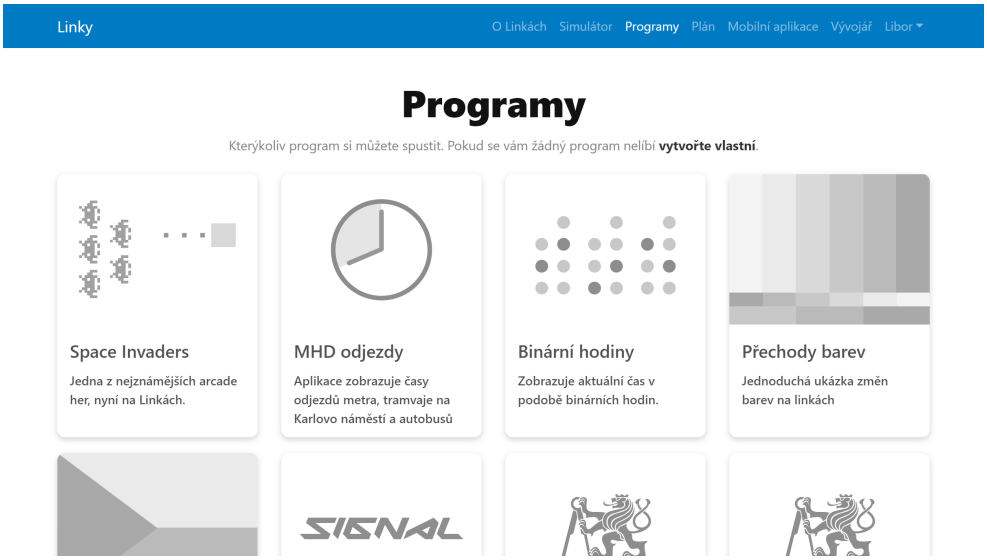
### Serverová aplikace

Serverovou aplikací se označuje aplikace, která je přidaná do informačního systému, která jde spustit a která dokáže vykreslovat data na světelné fasádě pomocí grafické knihovny. Serverová aplikace může být čistě animační, nebo může zobrazovat libovolný obraz na základě externích dat. Na fasádě tak mohou aplikace zobrazovat informace o počasí, informace o znečištění ovzduší, informace o odjezdech MHD v Dejvicích a neomezený počet dalších informací a jiných externích dat. Externími daty mohou například být i data odesílaná

přímo z mobilních zařízení, takže serverové aplikace mohou na fasádu vykreslovat animace nebo dokonce i hry, kde hlavními aktéry jsou přímo uživatelé systému stojící fyzicky před samotnou světelnou fasádou. Díky jednoduchosti a nezávislosti na konkrétní technologii může serverové aplikace vytvářet každý průměrný vývojář.

### Webový portál

Dalším systémem je webový portál, který slouží k ovládání systémového jádra a tím pádem i celého informačního systému. Přes webový portál se mohou do systému přihlásit správci systému a spravovat tak například běh serverových aplikací nebo ostatní uživatele. Dále se může přes webový portál do systému přihlásit vývojář aplikací, který může přidávat do systému nové serverové aplikace. Aplikace po přidání do systému automaticky čekají na schválení správcem systému. Během této doby je může vývojář i libovolný správce systému testovat na dostupném webovém simulátoru.



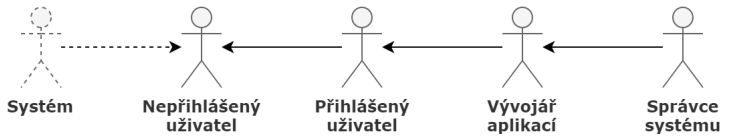

Obrázek 3.2: Ukázka nově vyvíjeného webového portálu [20]

### Grafická knihovna

Ve stávajícím informačním systému lze vykreslovat obraz pouze pomocí nastavení barvy a jasu všem jednotlivým bodům obrazu světelné fasády. Aby se serverovým aplikacím zjednodušilo vykreslování prvků na světelnou fasádu, vznikla komponenta grafická knihovna, která umožňuje vykreslovat na fasádu nejrůznější objekty (obdélník, čtverec, kruh, atp.). Těmto objektům lze nastavit barvu, jas, průhlednost a mohou se i navzájem překrývat [21].

## 3.3 Uživatelské role

Jako v každém informačním systému i v tomto se používají uživatelské role. Každému uživateli je přidělena právě jedna z následujících uživatelských rolí. Tyto role jsou si vždy konkrétním způsobem nadřazené. Na obrázku 3.3 je znázorněná hierarchie uživatelských rolí. Jádro systému je navrženo tak, aby bylo jednoduché přidat libovolnou další uživatelskou roli, avšak není k tomu navržen žádný API endpoint. Přidat novou uživatelskou roli je třeba manuálně ve zdrojovém kódu jádra systému.




Obrázek 3.3: Hierarchie uživatelských rolí nového systému

### Nepřihlášený uživatel

Nepřihlášený uživatel se nijak neautorizuje, nepotřebuje nic speciálního. Takovému uživateli je umožněno pouze zobrazení nejzákladnějších dat a přihlášení se do systému.

### Přihlášený uživatel

Uživatelská role *přihlášený uživatel* je nadřazená uživatelské roli *nepřihlášený uživatel* (co je umožněno nepřihlášenému uživateli, je umožněno i přihlášenému uživateli). Přihlášenému uživateli je pak navíc umožněno získání více informací, připojovat se k serverovým aplikacím, hrát hry nebo odhlásit se.

### Vývojář aplikací

Uživatel s rolí *vývojář aplikací* je nadřazený roli *přihlášený uživatel*. Vývojář aplikací může navíc přidávat do systému nové serverové aplikace a má možnost je editovat a testovat.

### Správce systému

Nejvyšší uživatelská role je správce systému. Správce systému může spravovat jednotlivé uživatele, mobilní zařízení i serverové aplikace. Může nastavovat a řídit celý informační systém, běh serverových aplikací či spravovat testovací prostředí.

### Systém

Aby se zachovala jednotnost a přehlednost celého informačního systému, existuje speciální uživatelská role systém, která označuje přístupy ostatních druhů systémů, které nejsou spojeny s žádným fyzickým uživatelem. Za systém lze považovat přístupy běžných systémů, přístupy serverových aplikací nebo přístupy grafické knihovny. Toto je ale pouze předpřipravená uživatelská role a v současné době ji nevyužívá žádná komponenta.

## 3.4 Případy užití

Případy užití přímo navazují na funkční požadavky nového řešení z předchozí kapitoly. U každého případu užití je uveden identifikátor, název, seznam primárních uživatelů, cíl a jeho předpoklady. Dále ještě může být případ užití doplněn krátkou poznámkou.

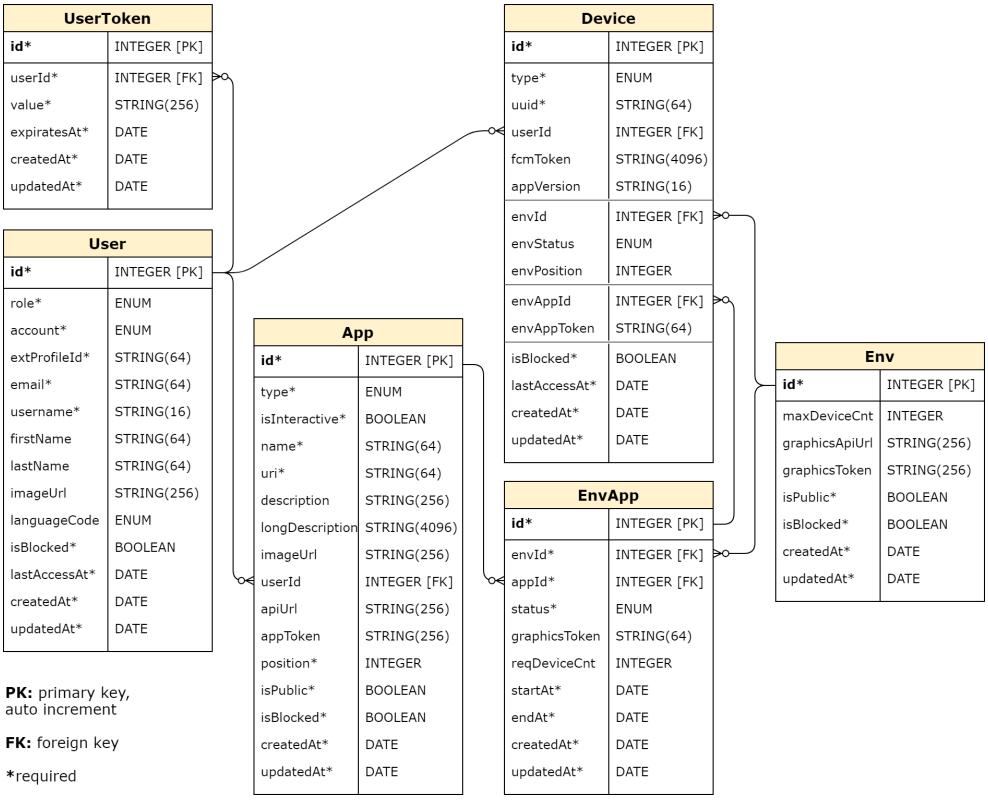
Ačkoliv jsou případy užití jednou z nejzákladnějších částí této práce, jsou kvůli své obsáhlosti umístěny v příloze C, na konci této práce. V této příloze je uveden kompletní, zcela vyčerpávající seznam všech případů užití, které jsou implementovány v praktické části této práce. Vzhledem k množství a obsáhlosti jsou tyto případy užití ještě navíc rozděleny do jednotlivých podkategorií. Je vhodné si tyto případy užití nejprve pročíst, před samotným pokračováním čtení následujících částí této práce.

## 3.5 Databáze

Tato sekce obsahuje návrh databáze. Vzhledem k jednoduchosti databáze je v této práci uveden pouze výčet jednotlivých databázových tabulek a databázový model.

Výčet všech databázových tabulek se nachází na konci této práce v příloze D. U každé tabulky je uveden název, seznam atributů a jejich typ, vlastnosti a popis. Každá tabulka obsahuje číselný primární klíč *id*, který je automaticky generovaný při vkládání nového datového objektu. U všech výčtových atributů se vždy nachází kompletní seznam možností, kterých mohou nabývat. U všech textových atributů je vždy uveden maximální počet znaků, který do nich lze uložit a do všech časových atributů lze uložit časové hodnoty s přesností na jednotky mikrosekund.

Databázový model se nachází na obrázku 3.4 a jednoduše zobrazuje, propojuje a shrnuje seznam všech databázových tabulek. Relace těchto tabulek (jejich primárních a cizích klíčů) jsou znázorněny obecně známou *Crow's Foot* [22] notací. Každý datový objekt je před uložením do databáze pečlivě ověřen. Ověřují se nejen základní věci, ale například i správnost URL adres.




Obrázek 3.4: Databázový model

## 3.6 API

Tato sekce obsahuje návrh rozhraní. Je zde popsáno vytváření dokumentace, jaké požadavky jádro systému podporuje a v jakém formátu vrací odpovědi.

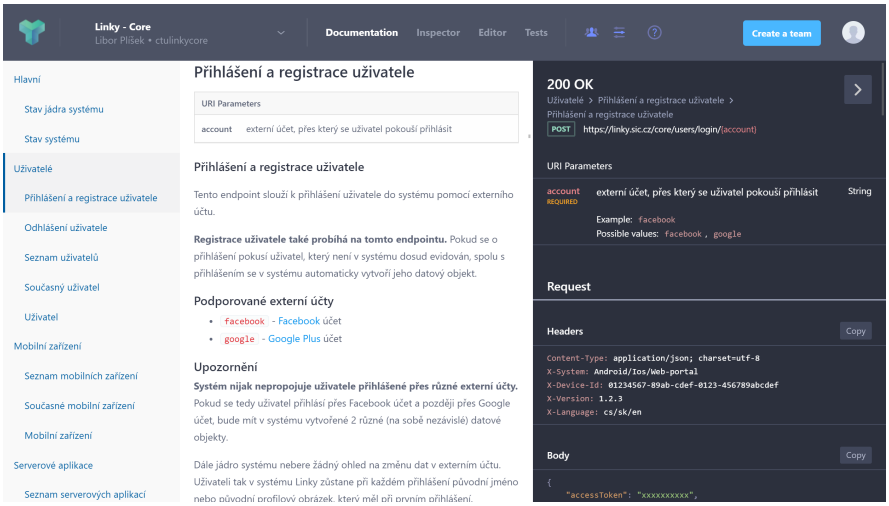
### Dokumentace

API dokumentace byla vyhotovena v online platformě *apiary*. [23] Nový informační systém je poměrně velký a je velmi pravděpodobné, že při budoucím vývoji, na něm bude zároveň spolupracovat více vývojářů. Tato platforma je velmi vhodná pro tvorbu online ve vzájemné spolupráci, a právě proto byla zvolena, pro tvorbu API dokumentace jádra nového informačního systému.

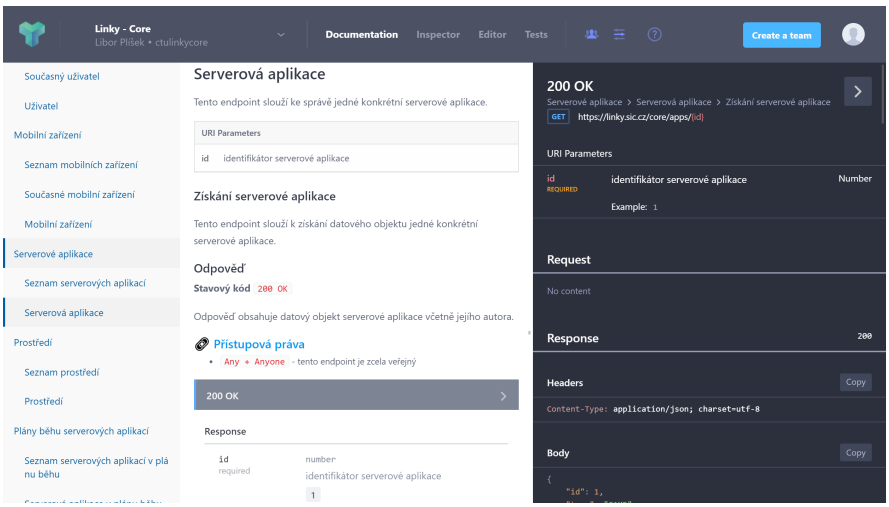
Nová API dokumentace je strukturovaná a obsahuje veškeré potřebné informace pro pochopení funkčnosti jádra systému. Tato dokumentace byla kompletně navržena v rámci této bakalářské práce a je dostupná online na adrese `https://ctulinkycore.docs.apiary.io/`. Zdrojový kód této dokumentace je psán ve formátu *API Blueprint* [24] a nachází se v elektronické příloze této práce.

Každý API endpoint obsažený v dokumentaci byl navržen přesně podle případů užití uvedených v příloze C této práce.

Nalevo se v online dokumanteci nachází rozcestník, uprostřed seznam API endpointů a jejich podrobnosti a napravo vždy konkrétní detail aktuálně označeného API endpointu. Na obrázcích 3.5 a 3.6 jsou ilustrace dvou konkrétních endpointů API dokumentace.




Obrázek 3.5: Ilustrace API dokumentace – přihlášení uživatele




Obrázek 3.6: Ilustrace API dokumentace – získání serverové aplikace

### Požadavky

Jádro systému přijímá HTTP [25] požadavky pouze v JSON formátu [26] a UTF-8 kódování [27]. Všechny ostatní požadavky, mohou být buď odmítnuty nebo zpracovány chybně. Veškeré časové údaje pak jádro systému přijímá pouze v ISO 8601 formátu [28].

```
GET /core/apps?limit=10&amp; offset=0 HTTP/1.1
Host: linky.sic.cz
X-System: Web-portal
X-Language: cs
X-Token:
User-Agent: PostmanRuntime/7.11.0
Accept: */*
Cache-Control: no-cache
Postman-Token: b4856e2d-a666-489e-bb2b-92569bbc3c92,c7a8da6a-6a62-45ee-b762-9a5b8de1947c
Host: linky.sic.cz
accept-encoding: gzip, deflate
Connection: keep-alive
cache-control: no-cache

```

Obrázek 3.7: Příklad HTTP požadavku na endpoint jádra systému

#### Přijímané metody

Jádro systému přijímá 4 následující HTTP metody. U každé metody je upřesněno, k čemu slouží, nebo na co se používá.

- **GET** – slouží k získání dat
- **POST** – slouží k vytvoření nového datového objektu
- **PUT** – slouží k úpravám existujícího datového objektu
- **DELETE** – slouží k odstranění existujícího datového objektu

#### Autorizace

Každý uživatel, který podléhá autorizaci se identifikuje autorizačním tokenem, který pošle v hlavičce HTTP dotazu. Každý systém má přidělený svůj identifikační název, který se taktéž odesílá v hlavičce HTTP dotazu. Jádro systému tak může přesně identifikovat uživatele a jeho práva, i z jakého druhu systému na API endpoint přistupuje. Veškeré podrobnosti o autorizaci jsou v úvodu nové online API dokumentace.

#### Rozšířené dotazování

Jádro systému podporuje rozšířené dotazování. Na endpointech určených k hromadnému načítání datových objektů z databáze, lze objekty řadit, stránkovat, filtrovat, a nebo v nich vyhledávat, a to podle nejrůznějších atributů. Všechny dostupné parametry a povolené atributy jsou vypsané a blíže specifikované zvlášť u jednotlivých endpointů v nové API dokumentaci.

## Odpovědi

Stejně tak jako požadavky i HTTP odpovědi jádra systému jsou pouze v JSON formátu a UTF-8 kódování, a zároveň i veškeré časové údaje jsou v ISO 8601 formátu.

Tyto odpovědi jádra systému se dále v této práci dělí na *standardní odpovědi* a na *chybové odpovědi*.

### Standardní odpovědi

Standardní odpovědi, používá jádro systému v případě, kdy došlo ke správnému dokončení požadované operace.

Následuje seznam všech typů standardních odpovědí, které vrací jádro informačního systému. U každé odpovědi je obsažen číselný stavový kód, název (napsaný kurzivou) a krátký popis.

### Seznam typů standardních odpovědí

- **Status 200 *OK*** – požadavek byl úspěšný
- **Status 201 *Created*** – požadavek byl úspěšný a byl vytvořen nový datový objekt
- **Status 204 *No Content*** – požadavek byl úspěšný (prázdná odpověď)


```
{
    "limit": 10,
    "offset": 0,
    "total": 9,
    "results": [
        {
            "id": 6,
            "type": "GAME",
            "isInteractive": false,
            "name": "Space Invaders",
            "uri": "space_invader",
            "description": "Jedna z nejznámějších arcade her, nyní na Linkách.",
            "longDescription": "Arcade hra space invader. Hráč je na linkách označen tmavě zelenou barvou. Nepřátelé červenou
                .\\n## Ovládání\\nLevé a pravé tlačítko slouží k pohybu do stran\nProstřední tlačítko střílí projektily.\n\n##
                Pravidla hry\nÚkolem hráče je sestřelit všechny nepřátele nežli se dostanou za hráče.",
            "imageUrl": "https://linky.sic.cz/uploads/af9d67d196411b53563c660ce627aaee.svg",
            "author": {
                "id": 1,
                "role": "ADMINISTRATOR",
                "username": "PanSpravce",
                "imageUrl": "https://graph.facebook.com/v2.6/10210217516305664/picture?type=large",
                "languageCode": "CS",
                "createdAt": "2019-04-12T10:53:23.477Z"
            },
            "createdAt": "2019-04-23T16:44:54.313Z"
        },
```


Obrázek 3.8: Příklad odpovědi se stavovým kódem 200 *OK*

### Chybové odpovědi

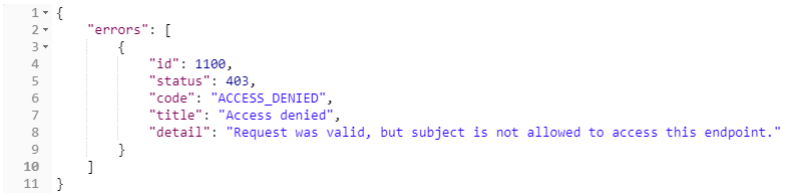
Chybové odpovědi mohou být vráceny v okamžiku, kdy se nepodaří dokončit požadovanou operaci. Všechny chybové odpovědi jsou v *JSON Error Objects* formátu [29].

Následuje kompletní seznam všech typů chybových odpovědí, které vrací jádro informačního systému. U každé odpovědi je obsažen číselný stavový kód, název (napsaný kurzivou) a krátký popis.

Výčet úplně všech možných chybových odpovědí (přes 130 záznamů), včetně jejich identifikátorů, stavových kódů, názvů, nadpisů i detailů, obsahuje nově navržená online API dokumentace.

**Seznam typů chybových odpovědí**

- **Status 400 *Bad Request*** – požadavek byl chybný, nebo chyběly požadované parametry
- **Status 401 *Unauthorized*** – požadavek byl chybný, nebo se nezdařilo ověření uživatelské role
- **Status 403 *Forbidden*** – ověření uživatelské role bylo úspěšné, ale uživatel nemá dostatečná práva pro vykonání požadované operace
- **Status 404 *Not Found*** – endpoint nebyl nalezen
- **Status 409 *Conflict*** – požadavek nebyl úspěšný, protože došlo ke konfliktu (konflikt vzniká například u neunikátního username)
- **Status 5xx *Server error*** – chyba na straně serveru (kde $x$ reprezentuje libovolné kladné celé číslo)



```
{
    "errors": [
        {
            "id": 1100,
            "status": 403,
            "code": "ACCESS_DENIED",
            "title": "Access denied",
            "detail": "Request was valid, but subject is not allowed to access this endpoint."
        }
    ]
}
```


Obrázek 3.9: Příklad odpovědi se stavovým kódem 400 *Bad request*

## 3.7 Démoni

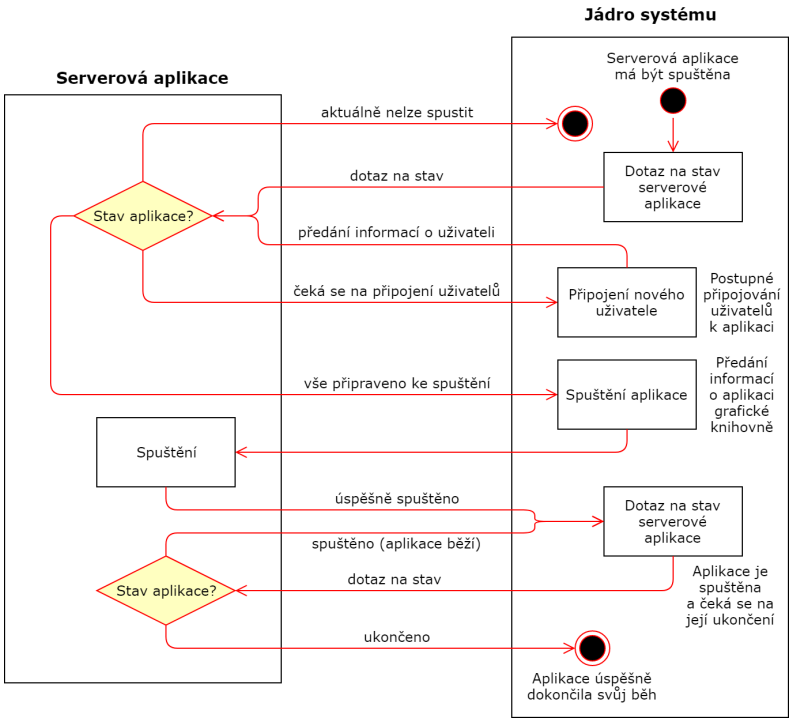
*„Démon je typ programu […], který běží nenápadně na pozadí (nikoli pod přímým ovládáním uživatele), a čeká, až bude aktivován výskytem určité události nebo podmínky.“* [30]

Jádro systému obsahuje dva základní démony. První zajišťuje správný běh aplikací v plánech běhu a druhý zajišťuje obsluhu mobilních zařízení ve frontách.

### Plán běhu serverových aplikací

První démon obsluhuje plány běhu serverových aplikací ve všech aktivních prostředích. Automaticky spouští a ukončuje jednotlivé serverové aplikace a posílá grafické knihovně informace o právě spuštěné serverové aplikaci.

Na obrázku 3.10 se nachází diagram spouštění serverových aplikací v plánech běhu. Každá aplikace má svůj definovaný životní cyklus. Jádro systému se postupně během spouštění aplikace dotazuje na její stav, čímž zjišťuje, v jaké fázi životního cyklu se aplikace nachází. Nejprve se jádro systému vždy snaží aplikaci spustit, interaktivní aplikace nejsou před spuštěním ihned připraveny, protože nejprve čekají na připojení uživatelů. Pokud je právě spouštěná aplikace interaktivní, jádro systému jí uživatele automaticky ihned přiřadí. Čistě animační a jiné neinteraktivní aplikace jsou připravené ke spuštění ihned. Jakmile je aplikace připravena ke spuštění, jádro systému pošle informaci o právě spouštěné aplikaci grafické knihovně a aplikaci příkaz, že se již má spustit. Následně je aplikace spuštěna a už se jen pravidelně kontroluje její stav a následné ukončení.

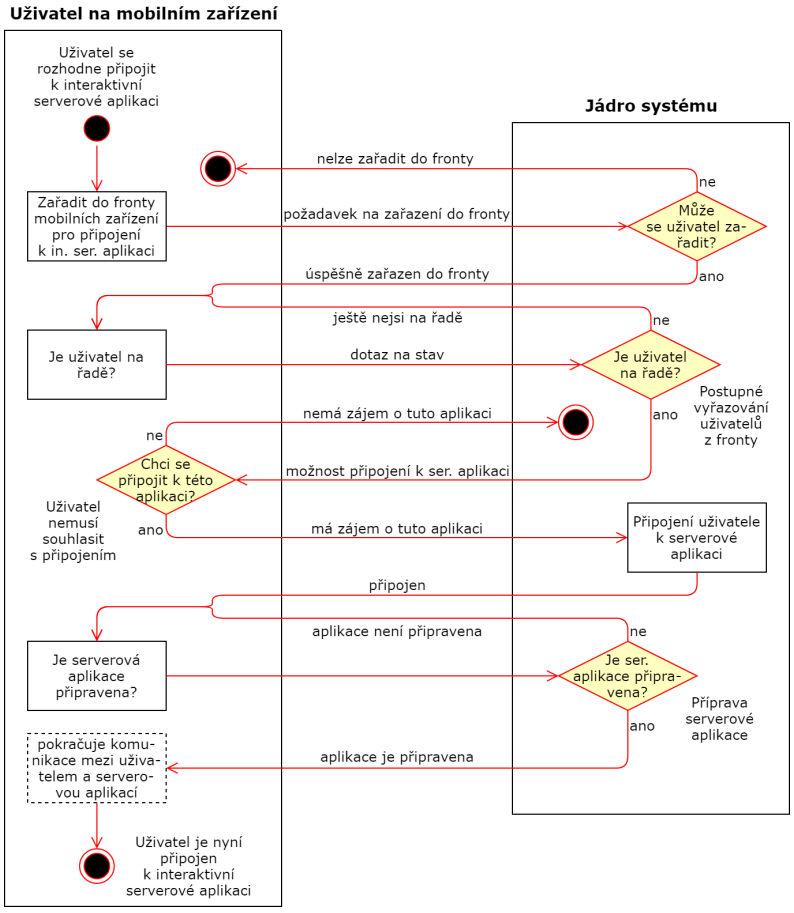



Obrázek 3.10: Diagram spouštění serverových aplikací

### Fronta mobilních zařízení

Druhý démon obsluhuje fronty mobilních zařízení ve všech aktivních prostředích. Automaticky připojuje mobilní zařízení k serverovým aplikacím a udržuje ve frontě pouze aktivní uživatele.

Na obrázku 3.11 je znázorněn diagram obsluhy mobilních zařízení ve frontách. Nejprve uživatel odešle přes své mobilní zařízení požadavek na zařazení do fronty. Poté vyčkává, než na něj vyjde řada. Pokud vyšla na uživatele řada, má uživatel možnost se připojit k serverové aplikaci. Uživatel se může rozhodnout, že se k serverové aplikaci nepřipojí (o právě spouštěnou serverovou aplikaci nemusí mít zájem). V takovém případě je automaticky vyřazen z fronty a na řadě je další uživatel. Po připojení všech potřebných uživatelů k serverové aplikaci je aplikace automaticky spuštěna. Po spuštění serverové aplikace uživatelé přes svá mobilní zařízení komunikují už pouze jenom se samotnou serverovou aplikací.




Obrázek 3.11: Diagram obsluhy mobilních zařízení

Kapitola 4

# Realizace

V této kapitole se nachází použité technologie, postup implementace navrženého řešení a následně postup nasazení implementovaného řešení na produkční server.

## 4.1 Použité technologie

Zadání bakalářské práce je poměrně striktní a nedává možnost volby použitých technologií. V této podkapitole následuje seznam základních technologií, které byly použity při realizaci jádra nového informačního systému. Jedná se pouze o výčet nejdůležitějších technologií. Použitých technologií je ve skutečnosti více, ale méně důležité technologie jsou průběžně zmiňovány v celé práci přímo u konkrétních případů, kterých se týkají.

### Node.js

*„Node.js je open source více-platformové prostředí pro vývoj serverových a síťových aplikací. Aplikace Node.js jsou napsány v jazyce JavaScript a lze je spustit v rámci běhového prostředí Node.js v operačních systémech OS X, Microsoft Windows a Linux.“* [31]

### Express.js

*„Express je minimální a flexibilní webový framework pro Node.js, který poskytuje robustní sadu funkcí pro webové a mobilní aplikace.“* [32]

### PostgreSQL

*„PostgreSQL je výkonný, open-source objektově-relační databázový systém s více než 30 lety aktivního vývoje, který si získal silnou reputaci pro spolehlivost, robustnost a výkon.“* [33]

### GitLab

*„GitLab je jediná aplikace pro celý životní cyklus vývoje softwaru. Od plánování projektů a řízení zdrojového kódu až po CI/CD, monitorování a bezpečnost.“* [34]

## 4.2 Implementace

Před samotným začátkem implementace softwarového jádra nového informačního systému, bylo třeba nejprve připravit vývojové prostředí. Po instalaci čistého linuxového serveru byly staženy a nainstalovány především programy *nodejs*, *postgresql* a *git*.

Po instalaci byl nejprve vytvořen Git repozitář. Původně byl pro tuto práci používaný GitLab repozitář na serveru gitlab.com, ale v průběhu této práce byl repozitář přesunut na školní GitLab server umístěný na adrese gitlab.fit.cvut.cz. Díky tomu mohl být projekt jednoduše sdílen s ostatními vývojáři nového informačního systému. Po založení verzovacího systému byla vytvořena databáze a následně pak mohl začít samotný vývoj jádra nového systému.

### Databáze

Databáze byla zprovozněna pomocí technologie *PostgreSQL*. Tato technologie nebyla zvolena z žádného konkrétního důvodu. Celkově není vůbec důležité, na jaké konkrétní technologii je postavena databáze, jádro systému ke správnému fungování potřebuje téměř libovolnou objektově relační databázi (ORD).

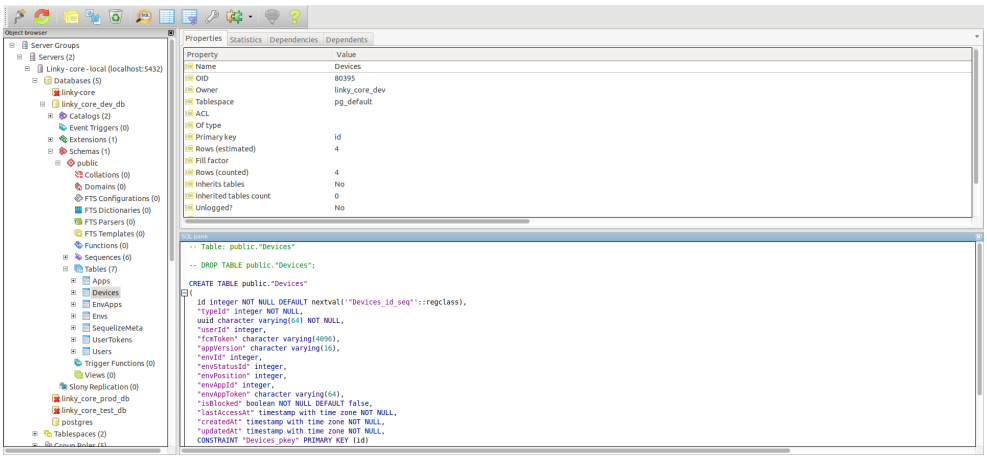
Po zprovoznění *PostgreSQL* dále byly vytvořeny 3 běhová prostředí jádra. Každému prostředí byla vytvořena vlastní databáze a vlastní přístupové údaje. Všechny přístupové údaje všech rozhraní se nakonec zanesly do konfiguračního souboru jádra nového informačního systému a databáze byla kompletně připravena k použití.

#### Seznam běhových prostředí

- Testovací – slouží k testování jádra systému
- Vývojové – slouží k vývoji a ladění jádra systému
- Produkční – slouží ke spuštění produkční verze jádra systému

Pro jednodušší obsluhu databáze byl nainstalován program *pgAdmin* [35], který obsahuje velmi jednoduché a intuitivní grafické uživatelské rozhraní (GUI) pro obsluhu databáze. Přes tento program se lze pomocí Secure Shell (SSH) tunelu připojit i přímo do databáze na produkčním serveru a spravovat tak reálná databázová data jádra informačního systému.

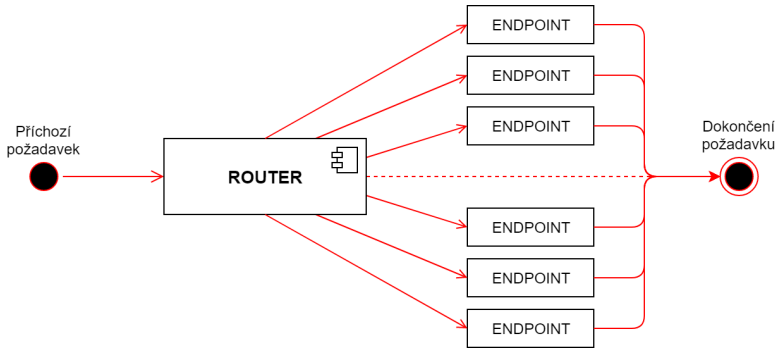
Pokud vývojář potřebuje změnit strukturu databáze jádra systému, změny lze nastavit v migračních souborech přímo ve zdrojovém kódu. Při odeslání nového řešení do online repozitáře se změny, po úspěšných testech, v databázi automaticky aplikují. Naopak při neúspěšných testech online repozitář automaticky kontaktuje vývojáře e-mailem a produkční řešení zůstane netknuté.



Obrázek 4.1: Ilustrace programu *pgAdmin*

## API

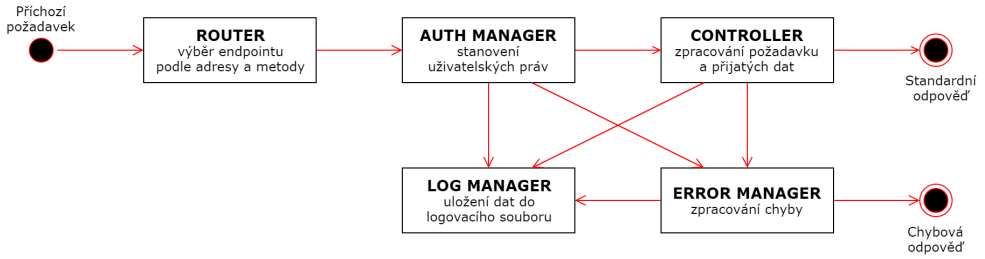
Základní prvkem implementace je router, který obsahuje cesty. Každá cesta představuje jeden konkrétní API endpoint. Implementovaný router tedy pomocí cest a HTTP [25] metod rozděluje a obsluhuje úplně celé REST API [6] jádra systému. Na obrázku 4.2 je znázorněna práce routeru. Jádrem systému jsou přijímány pouze požadavky v JSON formátu [26] a UTF-8 kódování [27]. Podrobnosti o formátu dotazů jsou obsaženy v předchozí kapitole. API bylo navrženo přesně podle případů užití nacházejících se v příloze C této práce.




Obrázek 4.2: Znázornění práce routeru

Každý požadavek je zpracován nejprve správcem autentizace (auth manager), kde se požadavek zkontroluje a stanoví se příslušná uživatelova práva, poté je požadavek zpracováván příslušným řadičem (controller) a při nesprávném požadavku proces dokončí správce chyb (error manager). O všechny záznamy v logovacích souborech se stará správce logování (log manager).

Na obrázku 4.3 je vyobrazeno zpracování obecného HTTP požadavku na REST API jádra systému.




Obrázek 4.3: Diagram zpracování obecného HTTP požadavku

## Použité balíčky

V této sekci je uveden seznam základníxh použitých externích balíčků a stručný popis k čemu se používají.

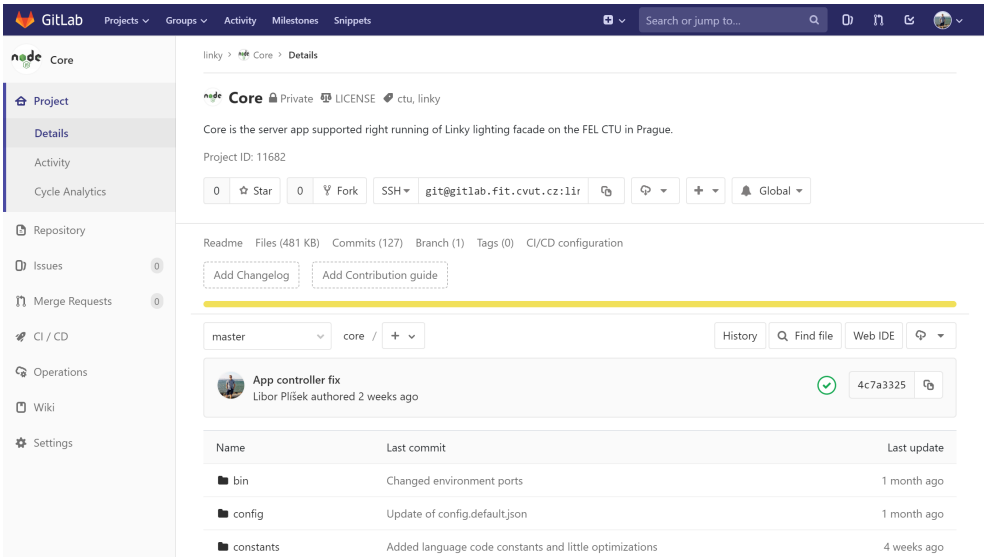
- debug – nástroj pro ladění
- express – již zmiňovaný Express.js framework
- morgan – HTTP logger middleware
- node-cron – malý plánovač úloh založený na GNU crontab
- passport-facebook-token – autentizační strategie tokenů pomocí Facebook OAuth 2.0 API
- passport-google-plus-token – autentizační strategie tokenů pomocí Google Plus OAuth 2.0 API
- qs – pro analýzu a rozbor HTTP požadavků
- request – nejjednodušší volání HTTP požadavků, podporuje i HTTPS a následuje přesměrování
- sequelize – Objektově relační zobrazení (ORM) pro objektově relační databáze (ORD)

## 4.3 Nasazení

Jak již bylo zmíněno pro vývoj jádra systému byl používán verzovací systém Git a to přes fakultní GitLab server. V tomto verzovacím systému byla zprovozněna průběžná integrace a nasazení (CI/CD) [18]. Při každé změně v online repozitáři, se nové řešení jádra systému ihned zkompiluje, otestuje a případně i automaticky nasadí na vzdáleném produkčním serveru. Pokud dojde k chybě, vývojář je automaticky upozorněn e-mailem a nové řešení se automaticky neaplikuje.

Veškeré uživatelské příručky obsahující návod k instalaci a návod ke spuštění jádra systému se nacházejí v elektronické příloze této práce.

Veškeré soubory se shellovými a CI/CD skripty použitých na GitLab a produkčním serveru jsou v elektronické příloze této práce.




Obrázek 4.4: Ukázka GitLab repozitáře

### Zálohování

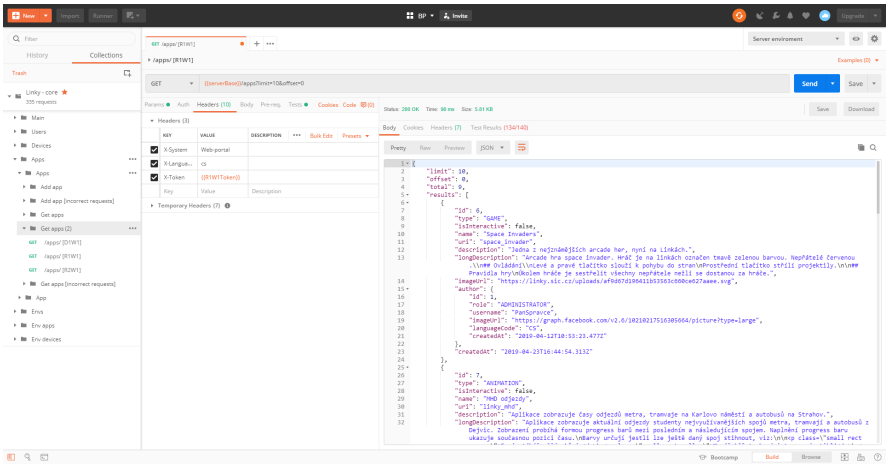
Databáze na produkčním serveru se zálohuje pravidelně každou 1 hodinu na pevný disk serveru. Pevný disk je automaticky zálohovaný každých 24 hodin poskytovatelem serveru. Zdrojový kód je automaticky zálohovaný verzovacím systémem Git v online repozitáři. Zálohováno je tedy do určité míry úplně vše. Při kompletním výpadku nebo odpojení serveru lze jádro systému spustit na libovolném jiném serveru s obnovenými daty starými maximálně jednu hodinu.

Kapitola 5

# Testování

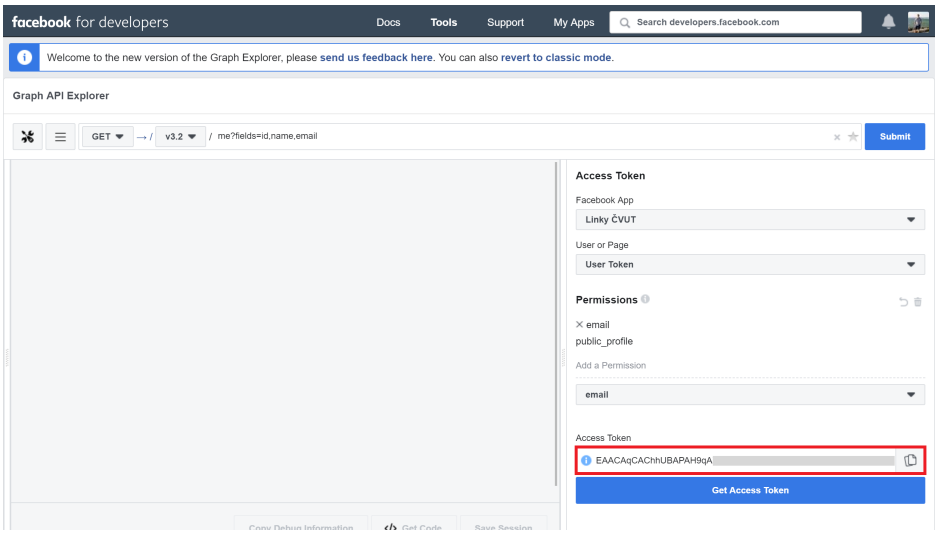
Při realizaci jádra nového informačního systému bylo použito API testování. Nebyly použity žádné jiné testy, dokonce ani zátěžové, protože tento typ serverové aplikace to nijak nevyžaduje. Jádro systému se nijak nepodílí přímo na vykreslování a streamování dat na světelnou fasádu, takže na výkon této aplikace nejsou kladeny žádné vysoké nároky. Nejdelší operace jsou tu pouze ty, když jádro systému odesílá požadavek po síti na nějakou jinou externí komponentu, například při přihlašování uživatelů přes externí Facebook nebo Google účet, nebo při spouštění serverové aplikace.

Pro vývoj, manuální i automatizované testování API byl použit jednoduchý nástroj *Postman* [36]. Postman obsahuje jednoduché intuitivní grafické uživatelské rozhraní (GUI), takže API testy se v něm vytvářejí velmi snadno. Lze si vytvořit různá testovací prostředí a v nich si lze nastavit i proměnné, takže testy na sebe mohou různě navazovat a chovají se tedy úplně stejně jako klientské aplikace.

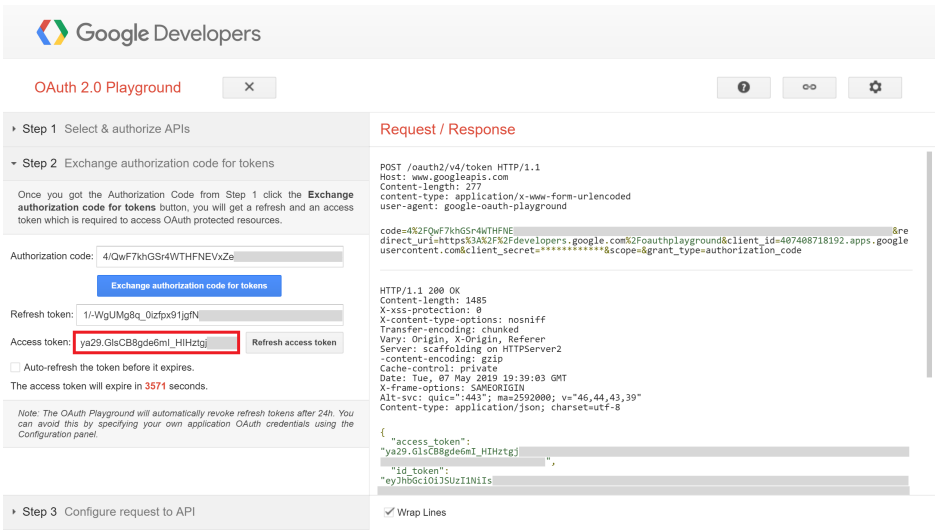


Obrázek 5.1: Illustrace programu Postman

Postman je napojen přímo na verzovací systém, takže při každé změně zdrojového kódu v online repozitáři se testy automaticky spustí a otestuje se nová verze řešení. Pro testování jsou použity i reálné uživatelské externí Google a Facebook účty, takže je možné testovat i přihlašování a odhlašování uživatelů i všechny uživatelské úrovně a požadavky. Před každým testováním je třeba vygenerovat a uložit do *Postman* prostředí aktuální uživatelské přístupové klíče vygenerované ve *Facebook Developers* a *Google Developers* rozhraních.



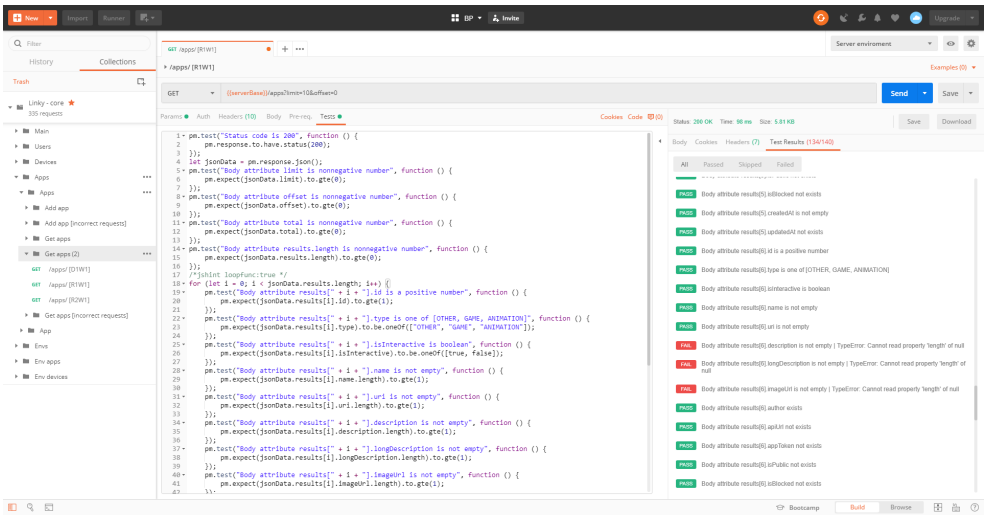

Obrázek 5.2: Příklad vygenerování klíče ve *Facebook Developers* rozhraní



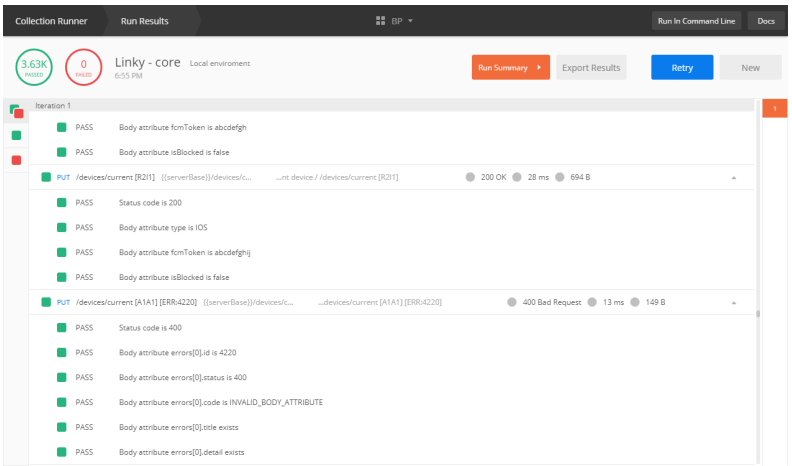

Obrázek 5.3: Příklad vygenerování klíče ve *Google Developers* rozhraní

Vytvořené testy pokrývají kompletně celou implementaci a byly vytvářeny v průběhu implementace praktické části této práce. Při testování se většinou objevily chyby hlavně v samotných testech, ale dohromady bylo objeveno i desítky chyb přímo v implementaci. Práce s vytvářením testů se tedy určitě vyplatila, všechny chyby objevené při testování byly vždy ihned opraveny.

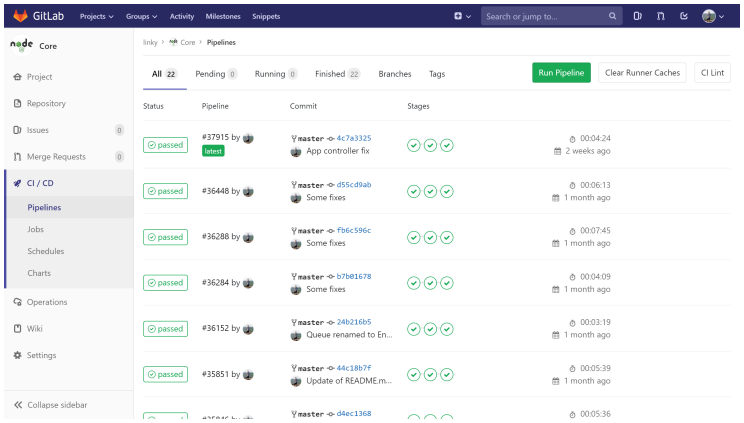
Zdrojové kódy API testů se nacházejí v elektronické příloze této práce.



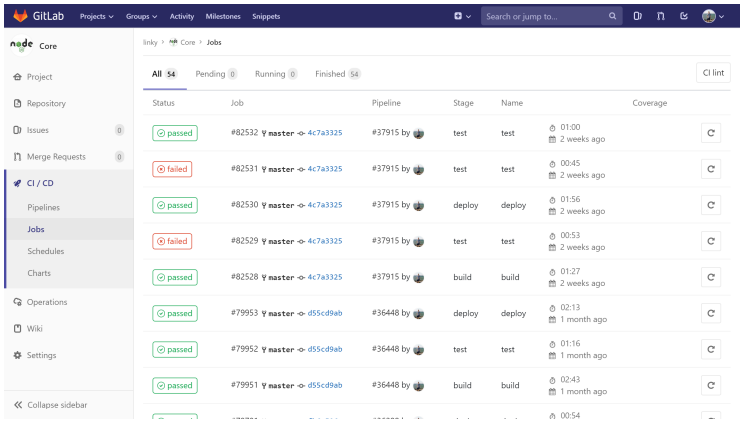
Obrázek 5.4: Ukázka manuálního testování v aplikaci *Postman*



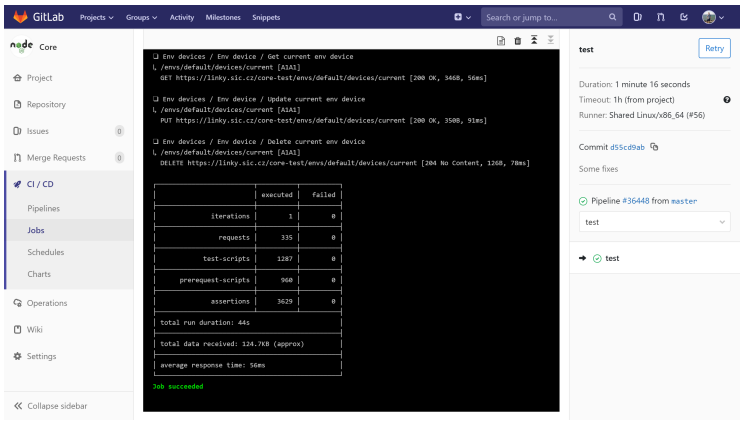

Obrázek 5.5: Ukázka automatizovaného testování v aplikaci *Postman*




Obrázek 5.6: Ukázka *Pipelines* v online repozitáři




Obrázek 5.7: Ukázka *Jobs* v online repozitáři




Obrázek 5.8: Ukázka automatizovaného testování v online repozitáři

# Závěr

Hlavním cílem této práce byla tvorba softwarového jádra, které umožňuje ovládání celého nového informačního systému, poskytuje přihlašování a správu uživatelů, správu a řízení serverových aplikací, správu a obsluhu klientských aplikací a zprostředkovává komunikaci mezi nimi.

Nejprve byla provedena analýza existujících řešení a analýza stávajícího i nového informačního systému. Při tvorbě analýzy bylo vycházeno z týmového projektu řešeného v rámci předmětů BI-SP1 a BI-SP2 ve druhém ročníku. Dále byly v rámci analýzy nového řešení informačního systému zpracovány funkční a nefunkční požadavky. Po analýze byl proveden návrh, při kterém byl navržen nejprve seznam typů systémů, seznam uživatelských rolí a případy užití. V případech užití bylo navrženo přihlašování a registrace uživatelů pomocí Google a Facebook účtů, návrh způsobu, jak obsluhovat mobilní zařízení, jak přidávat, odebírat, spouštět a ukončovat serverové aplikace, včetně automatického spouštění serverových aplikací podle předem nastaveného plánu. Poté byl vyhotoven databázový model a kompletní ucelený návrh komunikace mezi jádrem informačního systému, serverovými aplikacemi, mobilními zařízeními, webovým portálem, grafickou knihovnou a veřejnými systémy. Především byl vyhotoven důležitý návrh uceleného API, pomocí kterého lze softwarovému jádru posílat příkazy a tím ovládat celý nový informační systém. Dále v rámci návrhu byla vytvořena kompletní podrobná elektronická API dokumentace.

Po návrhu byla provedena implementace všech navržených řešení z teoretické části této bakalářské práce za pomocí technologie Node.js, průběžné integrace (continous integration) a verzovacího systému Git. Nejprve proběhlo seznamování se s uvedenými technologiemi, později byl vytvořen základní model databáze a první endpointy navrženého API. Časem se podařilo práci rozšiřovat a postupně přidat veškerou komunikaci a obsluhu všech komponent nového informačního systému včetně přihlašování uživatelů a plán pro automatické spouštění serverových aplikací. Vše bylo v průběhu implementace řádně testováno pomocí vhodných testů. Během testů byly objevovány a ihned napravovány veškeré chyby. Testy pokrývají kompletně celou implementaci.

Vzhledem k použití průběžné integrace, verzovacího systému a vytvoření podrobné dokumentace, je jádro připraveno k okamžitému budoucímu vývoji a rozšíření. Vše je připravené pro to, aby na jádru mohlo spolupracovat i více vývojářů zároveň. V budoucnosti by bylo možné systém obohatit především o nové serverové aplikace, které se přidají do nového informačního systému a poskytnou tak návštěvníkům nové interaktivní světelné zážitky ze světelné fasády. Dále by bylo vhodné rozšířit klientskou mobilní aplikaci také na platformu iOS, nebo vylepšit komunikaci mezi komponentami. Pro rychlejší obsluhu mobilních zařízení by bylo vhodné přidat do systému podporu push notifikací. Dále by bylo možné ukládat z odehraných herních zápasů také výsledky a jejich statistiky. Nebylo by na škodu také lepší zabezpečení serveru, například proti útokům na náhodné uhádnutí autentizačních API klíčů atp. Dále by bylo vhodné přidat do systému nastavení velikosti plátna v různých časových intervalech, nebo další uživatelské role. Nová uživatelská role by se mohla nazývat *tester*, a umožňovala by uživatelům připojovat se i k neveřejným serverovým aplikacím spuštěných v testovacích prostředích.

Všechny cíle této bakalářské práce byly úspěšně naplněny a softwarové jádro je připraveno (po vytvoření serverových a mobilních aplikací a následnému poslednímu odladění) k praktickému využití.

# Literatura


[1] Linky. In: ČVUT FEL – Linky [online]. ČVUT FEL IoT Workshop, 2018. [cit. 1. 4. 2019]. Dostupné z: https://linky.fel.cvut.cz/img/linky01.jpg

[2] BI-SP1 – Softwarový týmový projekt 1 [online]. ČVUT FIT. [cit. 9. 4. 2019]. Dostupné z: https://moodle.fit.cvut.cz/course/view.php?id=758

[3] BI-SP2 – Softwarový týmový projekt 2 [online]. ČVUT FIT. [cit. 9. 4. 2019]. Dostupné z: https://moodle.fit.cvut.cz/course/view.php?id=53

[4] CHLUDIL Jiří. Osobní konzultace o stávájícím informačním systému. Praha, 2018. [cit. 1. 4. 2019].

[5] About Node.js [online]. Node.js Foundation, 2019. [cit. 2. 4. 2019]. Dostupné z: https://nodejs.org/en/about/

[6] What is REST [online]. Roy Thomas Fielding, 2019. [cit. 2. 4. 2019]. Dostupné z: https://restfulapi.net/

[7] What is PHP? [online]. The PHP Group, 2019. [cit. 2. 4. 2019]. Dostupné z: https://www.php.net/manual/en/intro-whatis.php

[8] Pong – Videogame by Atari [online]. The International Arcade Museum®, 2019. [cit. 2. 4. 2019]. Dostupné z: https://www.arcade-museum.com/game_detail.php?game_id=9074

[9] Android [online]. Google, Inc., 2019. [cit. 4. 4. 2019]. Dostupné z: https://www.android.com/

[10] iOS [online]. Apple, Inc., 2019. [cit. 4. 4. 2019]. Dostupné z: https://www.apple.com/cz/ios/

[11] About Python [online]. Python Software Foundation, 2019. [cit. 2. 4. 2019]. Dostupné z: https://www.python.org/about

[12] ČVUT FEL – Linky [online]. ČVUT FEL IoT Workshop, 2018. [cit. 1. 4. 2019]. Dostupné z: https://linky.fel.cvut.cz/

[13] ČVUT FEL – Linky API [online]. ČVUT FEL IoT Workshop, 2018. [cit. 1. 4. 2019]. Dostupné z: https://linky.fel.cvut.cz/Api

[14] Seznamte se s funkcí Facebook Přihlášení [online]. Facebook, 2019. [cit. 10. 4. 2019]. Dostupné z: https://www.facebook.com/about/login

[15] Přihlašování na weby a do aplikací třetích stran pomocí účtu Google [online]. Google, 2019. [cit. 10. 4. 2019]. Dostupné z: https://support.google.com/accounts/answer/112802?hl=cs

[16] About Git [online]. Software Freedom Conservancy, 2019. [cit. 12. 4. 2019]. Dostupné z: https://git-scm.com/about

[17] The first single application for the entire DevOps lifecycle [online]. GitLab, 2019. [cit. 12. 4. 2019]. Dostupné z: https://about.gitlab.com/

[18] GitLab Continuous Integration & Delivery [online]. GitLab, 2019. [cit. 12. 4. 2019]. Dostupné z: https://about.gitlab.com/product/continuous-integration/

[19] Why HTTPS Matters [online]. Google Developers, 2019. [cit. 12. 5. 2019]. Dostupné z: https://developers.google.com/web/fundamentals/security/encrypt-in-transit/why-https

[20] Linky [online]. ČVUT, 2019. [cit. 1. 4. 2019]. Dostupné z: https://linky.sic.cz/

[21] Graphics library API 1.0 [online]. KOŠATA Jíří, 2019. [cit. 1. 4. 2019]. Dostupné z: http://185.88.73.77:5000/swagger

[22] Crow's Foot Notation [online]. University of Regina, 2008. [cit. 7. 5. 2019]. Dostupné z: http://www2.cs.uregina.ca/~bernatja/crowsfoot.html

[23] Powerful API Design Stack. Built for Developers. [online] Oracle and/or its affiliates., 2019. [cit. 14. 4. 2019]. Dostupné z: https://apiary.io/

[24] API Blueprint. [online] Oracle and/or its affiliates., 2019. [cit. 27. 4. 2019]. Dostupné z: https://apiblueprint.org/

[25] HTTP – Hypertext Transfer Protocol [online]. W3C®, 2003. [cit. 14. 4. 2019]. Dostupné z: https://www.w3.org/Protocols/HTTP/HTTP2.html

[26] Introducing JSON [online]. 2019. [cit. 14. 4. 2019]. Dostupné z: https://json.org/json-cz.html

[27] UTF-8 Encoding [online]. FileFormat.Info, 2019. [cit. 14. 4. 2019]. Dostupné z: https://www.fileformat.info/info/unicode/utf8.htm

[28] Date and time format – ISO 8601 [online]. Organisation internationale de normalisation, 2019. [cit. 14. 4. 2019]. Dostupné z: https://www.iso.org/iso-8601-date-and-time-format.html

[29] JSON:API – Specification v1.1 [online]. 2019. [cit. 14. 4. 2019]. Dostupné z: https://jsonapi.org/format/1.1/#error-objects

[30] Daemon Definition [online]. The Linux Information Project, 2005. [cit. 15. 4. 2019] (překlad vlastní). Dostupné z: http://www.linfo.org/daemon.html

[31] Node.js – Introduction [online]. Tutorials Point, 2019. [cit. 15. 4. 2019] (překlad vlastní). Dostupné z: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

[32] Express [online]. Node.js Foundation, 2017. [cit. 15. 4. 2019] (překlad vlastní). Dostupné z: https://expressjs.com/

[33] PostgreSQL [online]. The PostgreSQL Global Development Group, 2019. [cit. 15. 4. 2019] (překlad vlastní). Dostupné z: https://www.postgresql.org/

[34] The first single application for the entire DevOps lifecycle [online]. GitLab, 2019. [cit. 15. 4. 2019] (překlad vlastní). Dostupné z: https://about.gitlab.com/

[35] pgAdmin – PostgreSQL Tools [online]. 2019. [cit. 15. 4. 2019]. Dostupné z: https://www.pgadmin.org/

[36] Postman – Products [online]. Postman, 2019. [cit. 21. 4. 2019]. Dostupné z: https://www.getpostman.com/products

Příloha A

# Seznam použitých zkratek

**API** Application Programming Interface

**CD** Continuous delivery

**CI** Continuous integration

**FEL** Fakulta elektrotechnická

**FIT** Fakulta informačních technologií

**FR** Functional requirements

**GUI** Graphical User Interface

**HTTP** Hypertext Transfer Protocol

**HTTPS** Hypertext Transfer Protocol Secure

**ID** Identifikátor

**IS** Informační systém

**JSON** JavaScript Object Notation

**NR** Non-functional requirements

**ORD** Object-relational database

**ORDBMS** Object-relational database management system

**REST** Representational State Transfer

**SSH** Secure Shell

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

Příloha B

# Obsah přiloženého média

Přiloženým médiem k této práci je paměťová karta SDHC 16 GB.

- `readme.txt` ..... stručný popis obsahu paměťové karty
- `src` ..... zdrojové kódy
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `doc` ..... zdrojová forma dokumentace ve formátu API Blueprint
  - `test` ..... zdrojové kódy testování ve formátu JSON
  - `sh` ..... zdrojové kódy shellových skriptů
  - `img` ..... zdrojové kódy obrázkových příloh ve formátu draw.io
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `doc` ..... dokumentace
  - `api.txt` ..... odkaz na online API dokumentaci
- `img` ..... obrázkové přílohy práce

Příloha C

# Případy užití

### Hlavní

Následující sekce obsahuje hlavní případy užití. Všechny hlavní případy užití jsou dostupné pro širokou veřejnost (není zde vyžadována žádná autentizace).

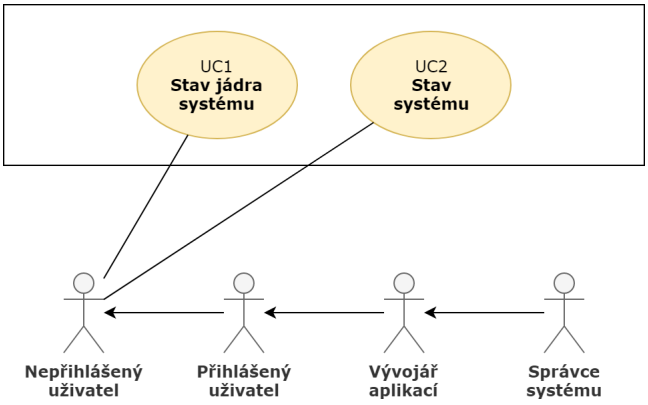
Na následujícím obrázku C.1 se nachází diagram hlavních případů užití.

#### UC1 – Stav jádra systému

- **Primární uživatel** – Nepřihlášený uživatel
- **Cíl** – Umožňuje získat základní informace o jádru systému, například: stav, URL adresu API dokumentace nebo systémový čas.
- **Předpoklady** – Tento případ užití je zcela veřejný (dostupný pro všechny bez ověření).

#### UC2 – Stav systému

- **Primární uživatel** – Nepřihlášený uživatel
- **Cíl** – Umožňuje získat základní informace o informačním systému Linky, například: stav světelné fasády (jestli je zapnutá nebo vypnutá), informace o právě spuštěné serverové aplikaci nebo předpokládaný čas spuštění další serverové aplikace.
- **Předpoklady** – Tento případ užití je zcela veřejný (dostupný pro všechny bez ověření).




Obrázek C.1: Hlavní případy užití

## Uživatelé

Následující sekce obsahuje případy užití související se správou a obsluhou uživatelů.

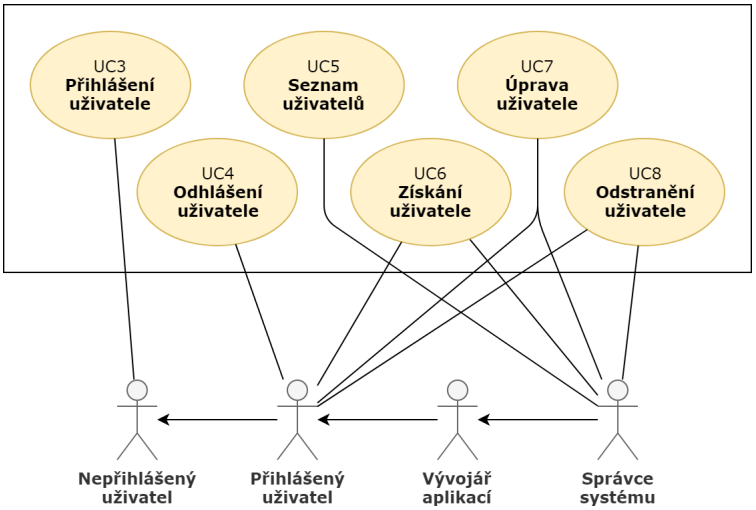
Případ užití *Registrace uživatele* není v novém informačním systému navržen, protože přihlašování je umožněno i neregistrovaným uživatelům. Uživatelé se mohou do systému přihlásit pouze pomocí externího účtu (Facebook nebo Google). Žádná jiná forma přihlášení není podporována.

Pokud se uživatel přihlásí do systému poprvé, automaticky se vytvoří jeho uživatelský účet a veškeré informace o uživateli jsou převzaté z právě použitého externího účtu. Uživatel je tak při prvním přihlášení v systému automaticky zaregistrovaný a díky tomu je případ užití pro registraci uživatele zcela nepotřebný.

Na následujícím obrázku C.2 se nachází diagram případů užití souvisejících s uživateli.

### UC3 – Přihlášení uživatele

- **Primární uživatel** – Nepřihlášený uživatel
- **Cíl** – Tento případ užití poskytuje přihlašování uživatelů do systému přes externí účet (Facebook nebo Google). Uživatelské účty se žádným způsobem neslučují, ke každému externímu účtu je vždy vytvořen právě jeden uživatelský účet. Pokud se uživatel přihlásí přes více externích účtů, bude v systému existovat více jeho uživatelských účtů. V jeden okamžik se může uživatel přihlásit na libovolném počtu prohlížečů i mobilních zařízení.
- **Předpoklady** – Uživatel musí přistupovat z podporovaného systému, například z mobilního zařízení nebo z webového portálu.




Obrázek C.2: Případy užití související s uživateli

**UC4 – Odhlášení uživatele**

- **Primární uživatel** – Přihlášený uživatel
- **Cíl** – Tento případ užití poskytuje odhlašování uživatelů ze systému.
- **Předpoklady** – Uživatel musí mít v systému evidovaný svůj účet a zároveň musí být již přihlášený na systému, ze kterého se chce odhlásit.

**UC5 – Seznam uživatelů**

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje získat evidované údaje o více uživatelských účtech naráz.
- **Předpoklady** – Uživatel vlastní příslušná práva pro získání seznamu uživatelských účtů.
- **Poznámka** – Implementace tohoto případu užití je realizována pomocí stránkování a jiných filtrů včetně vyhledávání.

**UC6 – Získání uživatele**

- **Primární uživatelé** – Přihlášený uživatel, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o uživateli. Přihlášený uživatel může získat základní informace o libovolném uživateli a zároveň kompletní evidované údaje o svém uživatelském účtu. Správce systému může získat kompletní informace o libovolném uživatelském účtu.

- **Předpoklady** – Uživatelský účet je evidován v systému a uživatel vlastní příslušná práva pro jeho získání.

#### UC7 – Úprava uživatele

- **Primární uživatelé** – Přihlášený uživatel, Správce systému
- **Cíl** – Umožňuje upravit evidované údaje o uživateli. Přihlášený uživatel může upravit informace pouze o svém uživatelském účtu. Správce systému může upravit evidovaná data o libovolném uživateli a také uživatele zablokovat, například pokud porušuje podmínky nebo zásady používání nového informačního systému.
- **Předpoklady** – Uživatelský účet je evidován v systému a uživatel vlastní příslušná práva pro jeho úpravu.

#### UC8 – Odstranění uživatele

- **Primární uživatelé** – Přihlášený uživatel, Správce systému
- **Cíl** – Umožňuje uživateli odstranit evidované údaje o uživatelském účtu. Správce systému může odstranit evidované údaje o libovolném uživateli. Přihlášený uživatel může odstranit pouze údaje o svém vlastním účtu. Odstranit účet ze systému se může hodit v případě, když není pravděpodobné nebo plánované jeho další použití.
- **Předpoklady** – Uživatelský účet je evidován v systému a uživatel vlastní příslušná práva pro jeho odstranění.
- **Poznámka** – Odstraněním uživatelského účtu ze systému nedojde k jeho zablokování.

### Mobilní zařízení

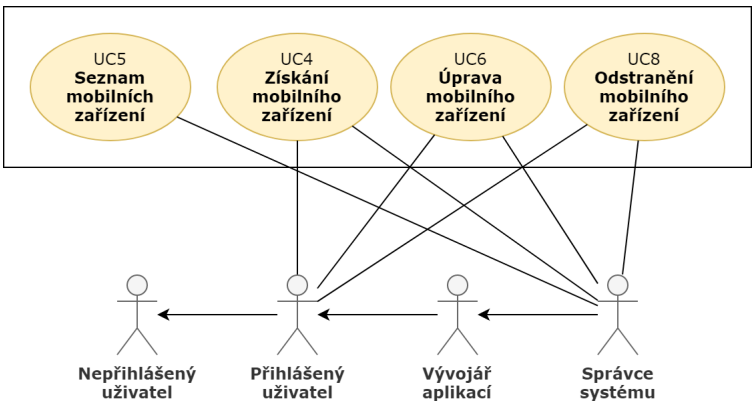
Následující sekce obsahuje případy užití související se správou a obsluhou mobilních zařízení. Přidání nového mobilního zařízení do systému probíhá plně automaticky bez vědomí uživatele. Jakmile k systému přistoupí mobilní zařízení, které není v systému dosud evidováno, systém automaticky uloží jeho data.

Na následujícím obrázku C.3 se nachází diagram případů užití souvisejících s mobilními zařízeními.

#### UC9 – Seznam mobilních zařízení

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje získat evidované údaje o více mobilních zařízení naráz.

- **Předpoklady** – Uživatel vlastní příslušná práva pro získání seznamu mobilních zařízení.
- **Poznámka** – Implementace tohoto případu užití je realizována pomocí stránkování a jiných filtrů.




Obrázek C.3: Případy užití související s mobilními zařízeními

### UC10 – Získání mobilního zařízení

- **Primární uživatelé** – Přihlášený uživatel, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o mobilním zařízení. Přihlášený uživatel může získat kompletní evidované údaje pouze o svém vlastním mobilním zařízení. Správce systému může získat kompletní informace o libovolném mobilním zařízení.
- **Předpoklady** – Mobilní zařízení je evidováno v systému a uživatel vlastní příslušná práva pro jeho získání.

### UC11 – Úprava mobilního zařízení

- **Primární uživatelé** – Přihlášený uživatel, Správce systému
- **Cíl** – Přihlášenému uživateli umožňuje upravit evidované údaje o svém mobilním zařízení. Tento případ užití může být využíván i samotnou mobilní aplikací bez přímého vědomí uživatele. Mobilní aplikace může samovolně aktualizovat token pro zasílání push notifikací a podobné systémové údaje. Správcům systému umožňuje upravit evidovaná data o libovolném mobilním zařízení a také mobilní zařízení zablokovat, například pokud porušuje podmínky nebo zásady používání nového informačního systému.

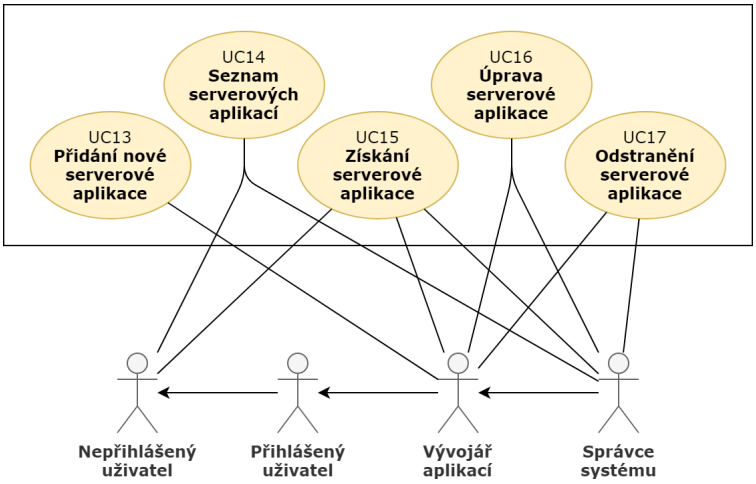
- **Předpoklady** – Mobilní zařízení je evidováno v systému a uživatel vlastní příslušná práva pro jeho úpravu.

### UC12 – Odstranění mobilního zařízení

- **Primární uživatelé** – Přihlášený uživatel, Správce systému
- **Cíl** – Umožňuje uživateli odstranit evidované údaje o mobilním zařízení. Správce systému může odstranit evidované údaje o libovolném zařízení. Přihlášený uživatel může odstranit údaje pouze o svém vlastním mobilním zařízení. Odstranit mobilní zařízení ze systému se může hodit v případě, kdy není pravděpodobné nebo plánované jeho další použití.
- **Předpoklady** – Mobilní zařízení je evidováno v systému a uživatel vlastní příslušná práva pro jeho odstranění.
- **Poznámka** – Odstraněním mobilního zařízení ze systému nedojde k jeho zablokování.

## Serverové aplikace

Následující sekce obsahuje případy užití související se správou a řízením serverových aplikací. Tato sekce neobsahuje případy užití spouštění a ukončení serverových aplikací. Serverové aplikace se spouští a ukončují plně automaticky démonem běžícím na pozadí serveru. Více o démonu obsluhujícím plány běhu serverových aplikací se lze dočíst v kapitole 3 této práce.




Obrázek C.4: Případy užití související se serverovými aplikacemi

**UC13 – Přidání nové serverové aplikace**

- **Primární uživatel** – Vývojář aplikací
- **Cíl** – Umožňuje přidat novou serverovou aplikaci do systému. Serverová aplikace nikdy není připravená k produkčnímu spuštění ihned po přidání do systému. Serverové aplikace musí před zveřejněním projít testovacím a schvalovacím procesem.
- **Předpoklady** – Uživatel vlastní příslušná práva pro přidání nové serverové aplikace.

**UC14 – Seznam serverových aplikací**

- **Primární uživatelé** – Nepřihlášený uživatel, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o více serverových aplikacích naráz. Správce systému může získat kompletní informace o všech serverových aplikacích. Nepřihlášený uživatel může získat pouze základní informace.
- **Předpoklady** – Tento případ užití je zcela veřejný (dostupný pro všechny bez ověření).
- **Poznámka** – Implementace tohoto případu užití je realizována pomocí stránkování a jiných filtrů včetně vyhledávání.

**UC15 – Získání serverové aplikace**

- **Primární uživatelé** – Nepřihlášený uživatel, Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o serverové aplikaci. Správce systému může získat kompletní informace o libovolné serverové aplikaci. Vývojář aplikací může získat kompletní informace pouze o své vlastní serverové aplikaci, kterou sám přidal do systému. Nepřihlášený uživatel může získat pouze základní informace o publikovaných aplikacích.
- **Předpoklady** – Serverová aplikace je evidována v systému. Tento případ užití je zcela veřejný (dostupný pro všechny bez ověření).

**UC16 – Úprava serverové aplikace**

- **Primární uživatelé** – Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje upravit evidované údaje o serverové aplikaci. Vývojář aplikací může omezeně editovat pouze vlastní serverovou aplikaci. Správce systému může kompletně upravovat evidovaná data libovolné serverové aplikace.

- **Předpoklady** – Serverová aplikace je evidována v systému a uživatel vlastní příslušná práva pro její úpravu.

#### UC17 – Odstranění serverové aplikace

- **Primární uživatelé** – Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje uživateli odstranit evidované údaje o serverové aplikaci. Správce systému může odstranit libovolnou serverovou aplikaci. Vývojář aplikací může odstranit ze systému pouze svoji vlastní aplikaci a pouze před schválením správcem systému. Odstranění serverové aplikace ze systému se může hodit v případě, když není pravděpodobné nebo plánované její další použití.
- **Předpoklady** – Serverová aplikace je evidována v systému a uživatel vlastní příslušná práva pro její odstranění.
- **Poznámka** – Odstraněním serverové aplikace ze systému dojde k jejímu zablokování. Pokud se vývojář aplikací rozhodne aplikaci do systému znovu přidat, správce systému ji musí před zveřejněním nejprve schválit.

### Prostředí

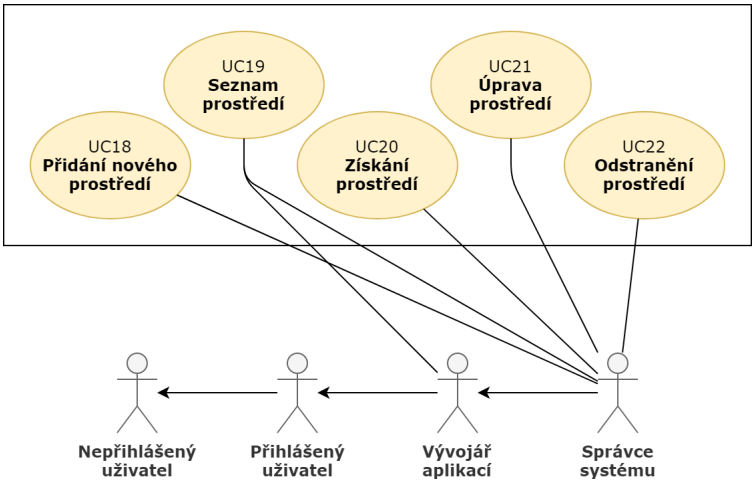
Následující sekce obsahuje případy užití související se správou prostředí.

Serverová aplikace po spuštění vždy odesílá grafická data přímo do grafické knihovny, ta pak na základě příkazů od jádra systému vykresluje data přímo na světelné fasádě nebo ve webovém simulátoru. Serverová aplikace tak není schopna rozpoznat, zda běží v produkčním, nebo v testovacím prostředí. Simulátor vykresluje grafická data naprosto stejným způsobem jako světelná fasáda. Díky tomuto řešení je možné serverové aplikace testovat v simulátoru přímo tak, jakoby právě běžely v produkčním prostředí.

Na následujícím obrázku C.5 se nachází diagram případů užití souvisejících s prostředími systému.

#### UC18 – Přidání nového prostředí

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje přidat nové produkční, nebo testovací prostředí do systému.
- **Předpoklady** – Uživatel vlastní příslušná práva pro přidání nového prostředí do systému.
- **Poznámka** – V systému vždy existuje právě jedno produkční a libovolný počet testovacích prostředí.




Obrázek C.5: Případy užití související s prostředími systému

### UC19 – Seznam prostředí

- **Primární uživatelé** – Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o více prostředích naráz. Správce systému může získat kompletní informace o všech prostředích. Vývojář aplikací může získat pouze základní informace.
- **Předpoklady** – Uživatel vlastní příslušná práva pro zobrazení seznamu prostředí.
- **Poznámka** – Implementace tohoto případu užití je realizována pomocí stránkování a jiných filtrů.

### UC20 – Získání prostředí

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje získat kompletní evidované údaje o prostředí.
- **Předpoklady** – Prostředí je evidováno v systému a uživatel má příslušná práva pro jeho zobrazení.

### UC21 – Úprava prostředí

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje upravit evidované údaje o prostředí.
- **Předpoklady** – Prostředí je evidováno v systému a uživatel má příslušná práva pro jeho úpravu.

- **Poznámka** – V systému vždy existuje právě jedno produkční prostředí.

**UC22 – Odstranění prostředí**

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje odstranit evidované údaje o prostředí.
- **Předpoklady** – Prostředí je evidováno v systému a uživatel má příslušná práva pro jeho odstranění.
- **Poznámka** – V systému existuje vždy právě jedno produkční prostředí, toto prostředí tedy nelze odstranit.

### Plány běhu serverových aplikací

Následující sekce obsahuje případy užití týkající se plánu běhu serverových aplikací.

Plán běhu existuje zvlášť pro každé evidované prostředí v systému. Obsahuje serverové aplikace a s nimi i časy jejich plánovaného spuštění a ukončení. Plán běhu obsahuje i historii běhu všech serverových aplikací.

Plán běhu serverových aplikací v produkčním prostředí je v této sekci dále nazýván jako *produkční plán běhu*. A stejně tak plán běhu v libovolném testovacím prostředí je v této sekci dále nazýván jako *testovací plán běhu*.

Démon obsluhující plán běhu serverových aplikací automaticky spouští nebo ukončuje serverové aplikace přesně podle předem nastaveného harmonogramu.

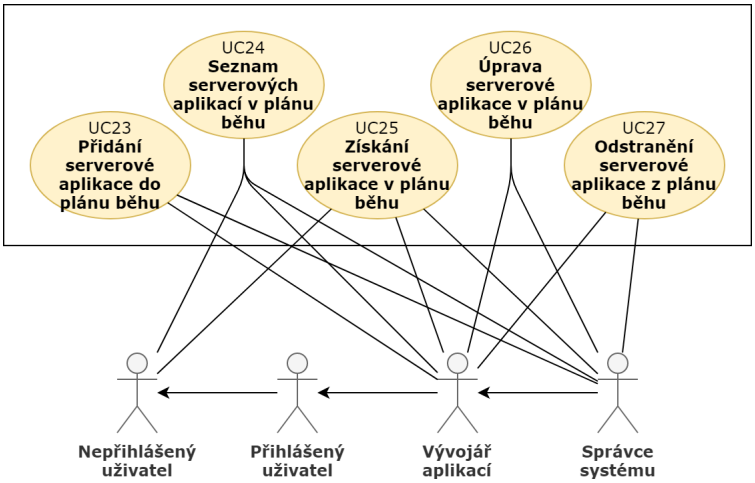
Žádné dvě různé serverové aplikace nemohou běžet v jednom prostředí ve stejný čas.

Žádná jedna serverová aplikace nemůže běžet vícekrát ve stejný čas (ani ve dvou různých prostředích).

Na následujícím obrázku C.6 se nachází diagram případů užití souvisejících s plány běhu serverových aplikací.

**UC23 – Přidání serverové aplikace do plánu běhu**

- **Primární uživatelé** – Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje přidat serverovou aplikaci do plánu běhu. Správce systému může přidat libovolnou serverovou aplikaci do libovolného plánu běhu. Vývojář aplikací může přidávat pouze vlastní serverové aplikace do testovacího plánu běhu.
- **Předpoklady** – Uživatel vlastní příslušná práva pro přidání serverové aplikace do plánu běhu.




Obrázek C.6: Případy užití související s plány běhu serverových aplikací

### UC24 – Seznam serverových aplikací v plánu běhu

- **Primární uživatelé** – Nepřihlášený uživatel, Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o více serverových aplikacích v plánu běhu naráz. Správce systému může získat kompletní informace o všech serverových aplikacích v libovolném plánu běhu. Vývojář aplikací může získat kompletní informace o svých vlastních serverových aplikacích a základní informace o všech ostatních aplikacích v libovolném plánu běhu. Nepřihlášený uživatel může získat pouze základní informace o všech serverových aplikacích v produkčním plánu běhu.
- **Předpoklady** – Uživatel vlastní příslušná práva pro zobrazení serverových aplikací v plánu běhu.
- **Poznámka** – Implementace tohoto případu užití je realizována pomocí výběru časových intervalů běhu aplikace a jiných filtrů.

### UC25 – Získání serverové aplikace v plánu běhu

- **Primární uživatelé** – Nepřihlášený uživatel, Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje získat evidované údaje o serverové aplikaci v plánu běhu. Správce systému může získat kompletní informace o libovolné aplikaci v libovolném plánu běhu. Vývojář aplikací může získat kompletní informace o své vlastní serverové aplikaci v libovolném plánu běhu. Nepřihlášený uživatel může získat pouze základní informace o libovolné serverové aplikaci v produkčním plánu běhu.

- **Předpoklady** – Serverová aplikace je evidována v plánu běhu a uživatel má příslušná práva pro její zobrazení.

#### UC26 – Úprava serverové aplikace v plánu běhu

- **Primární uživatelé** – Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje upravit evidované údaje, o serverové aplikaci v plánu běhu. Správce systému může upravovat libovolnou aplikaci v libovolném plánu běhu. Vývojář aplikací může upravovat evidovaná data pouze o vlastní serverové aplikaci v testovacím plánu běhu.
- **Předpoklady** – Serverová aplikace je evidována v plánu běhu a uživatel má příslušná práva pro její úpravu.

#### UC27 – Odstranění serverové aplikace z plánu běhu

- **Primární uživatelé** – Vývojář aplikací, Správce systému
- **Cíl** – Umožňuje odstranit evidované údaje o serverové aplikaci v plánu běhu. Správce systému může odstranit data libovolné serverové aplikace z libovolného plánu běhu. Vývojář aplikací může odstranit data pouze vlastní serverové aplikace z testovacího plánu běhu.
- **Předpoklady** – Serverová aplikace je evidována v plánu běhu a uživatel má příslušná práva pro její odstranění.

### Fronty mobilních zařízení

Následující sekce obsahuje případy užití související s frontou mobilních zařízení. Fronta mobilních zařízení existuje zvlášť pro každé evidované prostředí v systému. Obsahuje seznam mobilních zařízení připravených na připojení k interaktivní serverové aplikaci.

Fronta mobilních zařízení v produkčním prostředí je v této sekci dále nazývána jako *produkční fronta*. A stejně tak fronta mobilních zařízení v libovolném testovacím prostředí je v této sekci dále nazývána jako *testovací fronta*.

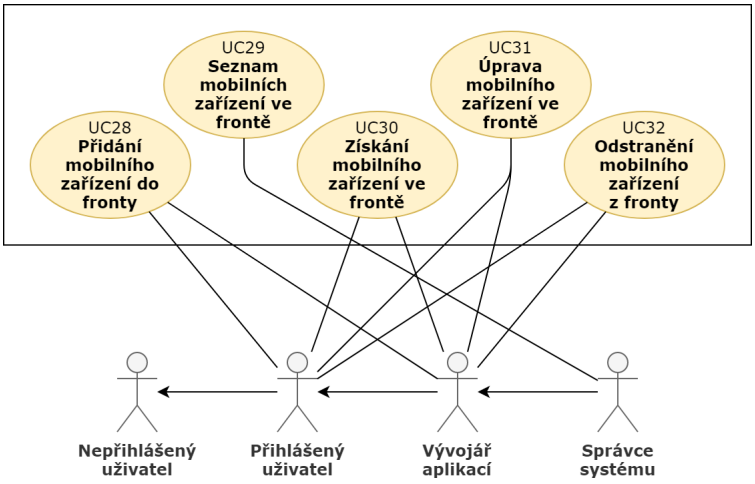
Démon obsluhující fronty mobilních zařízení automaticky umožňuje mobilním zařízením připojení k interaktivním serverovým aplikacím přesně podle jejich pořadí ve frontě.

Na následujícím obrázku C.7 se nachází diagram případů užití souvisejících s frontami mobilních zařízení.

#### UC28 – Přidání mobilního zařízení do fronty

- **Primární uživatelé** – Přihlášený uživatel, Vývojář aplikací

- **Cíl** – Umožňuje přidat uživatelovo mobilní zařízení do fronty. Vývojář aplikací může přidat své mobilní zařízení do libovolné fronty. Přihlášený uživatel může přidávat své zařízení pouze do produkční fronty.
- **Předpoklady** – Uživatel vlastní příslušná práva pro přidání svého mobilního zařízení do fronty.




Obrázek C.7: Případy užití související s frontami mobilních zařízení

### UC29 – Seznam mobilních zařízení ve frontě

- **Primární uživatel** – Správce systému
- **Cíl** – Umožňuje získat evidované údaje o více mobilních zařízeních ve frontě naráz. Správce systému může získat kompletní informace o všech mobilních zařízeních v libovolné frontě.
- **Předpoklady** – Uživatel vlastní příslušná práva pro zobrazení mobilních zařízení ve frontě.
- **Poznámka** – Implementace tohoto případu užití je realizována pomocí stránkování a jiných filtrů.

### UC30 – Získání mobilního zařízení ve frontě

- **Primární uživatelé** – Přihlášený uživatel, Vývojář aplikací
- **Cíl** – Umožňuje získat evidované údaje o mobilním zařízení ve frontě. Vývojář aplikací může získat evidované údaje o svém zařízení i z testovací fronty. Přihlášený uživatel může získat údaje o svém zařízení pouze z produkční fronty.

- **Předpoklady** – Mobilní zařízení je evidováno ve frontě a uživatel má příslušná práva pro jeho zobrazení.

**UC31 – Úprava mobilního zařízení ve frontě**

- **Primární uživatelé** – Přihlášený uživatel, Vývojář aplikací
- **Cíl** – Umožňuje upravit evidované údaje o mobilním zařízení ve frontě. Tento případ užití je primárně používán pro připojení mobilního zařízení k dostupné serverové aplikaci. Vývojář aplikací se může připojit k aplikaci i v testovacím prostředí. Přihlášený uživatel se může připojit k aplikaci pouze v produkčním prostředí.
- **Předpoklady** – Mobilní zařízení je evidováno ve frontě a uživatel má příslušná práva pro jeho úpravu.

**UC32 – Odstranění mobilního zařízení z fronty**

- **Primární uživatelé** – Přihlášený uživatel, Vývojář aplikací
- **Cíl** – Umožňuje odstranit evidované údaje o mobilním zařízení ve frontě. Vývojář aplikací může odstranit své mobilní zařízení z libovolné testovací fronty. Přihlášený uživatel může odstranit své mobilní zařízení pouze z produkční fronty.
- **Předpoklady** – Mobilní zařízení je evidováno ve frontě a uživatel má příslušná práva pro jeho odstranění.

Příloha D

# Databázové tabulky

| Název | Typ a vlastnosti | Popis |
|---|---|---|
| id* | INTEGER [PK] | Primární klíč datového objektu |
| role* | ENUM | Uživatelská role [nepřihlášený uživatel, přihlášený uživatel, vývojář aplikací, správce systému] |
| account* | ENUM | Typ externího účtu [Facebook, Google] |
| extProfileId* | STRING(64) | Identifikátor externího účtu |
| email* | STRING(64) | E-mail uživatele |
| username* | STRING(16) | Přezdívka uživatele |
| firstName | STRING(64) | Křestní jméno uživatele |
| lastName | STRING(64) | Příjmení uživatele |
| imageUrl | STRING(256) | Profilový obrázek uživatele |
| languageCode | ENUM | Kód jazyka [cs, sk, en] |
| isBlocked* | BOOLEAN | Je uživatel zablokovaný? |
| lastAccessAt* | DATE | Datum a čas posledního přístupu uživatele na server |
| createdAt* | DATE | Datum a čas vytvoření datového objektu v databázi |
| updatedAt* | DATE | Datum a čas poslední úpravy datového objektu v databázi |

Tabulka D.1: Databázová tabulka *User*

| Název | Typ a vlastnosti | Popis |
|---|---|---|
| id* | INTEGER [PK] | Primární klíč datového objektu |
| userId* | INTEGER [FK] | Identifikátor uživatele |
| value* | STRING(256) | Autentizační token uživatele |
| expiratesAt* | DATE | Datum a čas expirace tokenu |
| createdAt* | DATE | Datum a čas vytvoření datového objektu v databázi |
| updatedAt* | DATE | Datum a čas poslední úpravy datového objektu v databázi |

Tabulka D.2: Databázová tabulka *UserToken*

| Název | Typ a vlastnosti | Popis |
|---|---|---|
| id* | INTEGER [PK] | Primární klíč datového objektu |
| type* | ENUM | Typ zařízení [Android, iOS] |
| uuid* | STRING(64) | Univerzálně unikátní identifikátor zařízení |
| userId | INTEGER [FK] | Identifikátor uživatele |
| fcmToken | STRING(4096) | FCM token pro odesílání push notifikací do zařízení |
| appVersion | STRING(16) | Verze mobilní aplikace |
| envId | INTEGER [FK] | Identifikátor prostředí, ve kterém je zařízení přiřazeno do fronty |
| envStatus | ENUM | Stav zařízení ve frontě [čeká na připojení, je připojeno k aplikaci, …] |
| envPosition | INTEGER | Pozice zařízení ve frontě |
| envAppId | INTEGER [FK] | Identifikátor aplikace v prostředí, ke které je zařízení připojeno |
| envAppToken | STRING(64) | Autentizační token zařízení |
| isBlocked* | BOOLEAN | Je zařízení zablokované? |
| lastAccessAt* | DATE | Datum a čas posledního přístupu zařízení na server |
| createdAt* | DATE | Datum a čas vytvoření datového objektu v databázi |
| updatedAt* | DATE | Datum a čas poslední úpravy datového objektu v databázi |

Tabulka D.3: Databázová tabulka *Device*

| Název | Typ a vlastnosti | Popis |
|---|---|---|
| id* | INTEGER [PK] | Primární klíč datového objektu |
| type* | ENUM | Typ aplikace [animace, hra, …] |
| isInteractive* | BOOLEAN | Je aplikace interaktivní? |
| name* | STRING(64) | Název aplikace |
| uri* | STRING(64) | URI aplikace |
| description | STRING(256) | Krátký popis aplikace |
| longDescription | STRING(4096) | Dlouhý popis aplikace |
| imageUrl | STRING(256) | Náhledový obrázek aplikace |
| userId | INTEGER [FK] | Identifikátor autora aplikace |
| apiUrl | STRING(256) | API URL aplikace |
| appToken | STRING(256) | Autentizační token aplikace |
| position* | INTEGER | Pozice aplikace |
| isPublic* | BOOLEAN | Je aplikace dostupná pro veřejnost? |
| isBlocked* | BOOLEAN | Je aplikace zablokovaná? |
| createdAt* | DATE | Datum a čas vytvoření datového objektu v databázi |
| updatedAt* | DATE | Datum a čas poslední úpravy datového objektu v databázi |

Tabulka D.4: Databázová tabulka *App*

| Název | Typ a vlastnosti | Popis |
|---|---|---|
| id* | INTEGER [PK] | Primární klíč datového objektu |
| maxDeviceCnt | INTEGER | Maximální počet zařízení ve frontě |
| graphicsApiUrl | STRING(256) | API URL grafické knihovny |
| graphicsToken | STRING(256) | Autentizační token pro přístup ke grafické knihovně |
| isPublic* | BOOLEAN | Je prostředí dostupné pro veřejnost? |
| isBlocked* | BOOLEAN | Je prostředí zablokované? |
| createdAt* | DATE | Datum a čas vytvoření datového objektu v databázi |
| updatedAt* | DATE | Datum a čas poslední úpravy datového objektu v databázi |

Tabulka D.5: Databázová tabulka *Env*

| **Název** | **Typ a vlastnosti** | **Popis** |
|---|---|---|
| id* | INTEGER [PK] | Primární klíč datového objektu |
| envId* | INTEGER [FK] | Identifikátor prostředí |
| appId* | INTEGER [FK] | Identifikátor serverové aplikace |
| status* | ENUM | Stav serverové aplikace v plánu běhu [připravená na spuštění, čeká na zařízení, je spuštěná, …] |
| graphicsToken | STRING(64) | Autentizační token pro přístup ke grafické knihovně |
| reqDeviceCnt | INTEGER | Požadovaný počet mobilních zařízení |
| startAt* | DATE | Datum a čas plánovaného/reálného spuštění serverové aplikace |
| endAt* | DATE | Datum a čas plánovaného/reálného ukončení serverové aplikace |
| createdAt* | DATE | Datum a čas vytvoření datového objektu v databázi |
| updatedAt* | DATE | Datum a čas poslední úpravy datového objektu v databázi |

Tabulka D.6: Databázová tabulka *EnvApp*