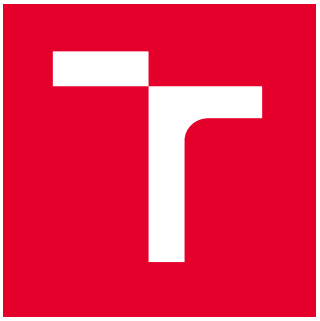


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

APLIKACE PRO DOBROVOLNÉ HASIČE

APPLICATION FOR VOLUNTEER FIRE DEPARTMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Matěj Vopálka**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Gerlich**

SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Matěj Vopálka ***ID:*** 220834

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Aplikace pro dobrovolné hasiče

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je návrh a implementace komplexní aplikace pro dobrovolné hasiče. Cílená aplikace bude ve formě mobilní aplikace a webové aplikace, které budou společně spolupracovat a splní následující funkcionalitu. Mobilní aplikace bude zobrazovat informace o výjezdu pomocí grafického uživatelského rozhraní a zda je člena hasičského sboru schopnen se výjezdu účastnit. Vytvořená mobilní aplikace bude komunikovat s webovým serverem (aplikace musí být provozována na nízkonákladovém HW př. raspberry pi nebo tablet), na kterém operátor uvidí akceschopnost jednotky na stanici. Serverová aplikace bude umožňovat základní správu uživatelských účtů a přehledné zobrazování dat v databázi. Bude zde také zachována historie výjezdu jednotky.

DOPORUČENÁ LITERATURA:

[1] ADLER, R.M. Distributed coordination models for client/server computing. Computer. 28(4), 14-22. ISSN 00189162. Dostupné z: doi:10.1109/2.375173

[2] GUAN, Mo a Minghai GU. Design and implementation of an embedded web server based on ARM. 2010 IEEE International Conference on Software Engineering and Service Sciences. IEEE, 2010, 2010, , 612-615. ISBN 978-1-4244-6054-0. Dostupné z: doi:10.1109/ICSESS.2010.5552275

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: Ing. Tomáš Gerlich

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práce se zabývá vývojem webové aplikace pro dobrovolné hasiče. Aplikace umožňuje členovi jednotky reagovat na vyhlášení výjezdu a zobrazuje stavy jednotlivých členů na monitoru v hasičské zbrojnici. Zajišťuje ověřování uživatelů a umožňuje zobrazení informací o výjezdu včetně účastnících se hasičů. Aplikace přijme informaci o výjezdu pomocí webového aplikačního rozhraní od Raspberry Pi, které čte a formátuje SMS zprávy.

KLÍČOVÁ SLOVA

Hasiči, JSDH, .NET, Android, HTTP, WebSocket, Webová aplikace, Webové push notifikace

ABSTRACT

The thesis deals with the development of a web application for volunteer firefighters. The application allows a firefighter to respond to the incident notification and display the status of individual members on the monitor in the fire station. The aplication provides user authentication and allows the viewing of information about the incident, including participating firefighters. The application receives incident information via the web application interface from Raspberry Pi, which reads and formats SMS messages.

KEYWORDS

Firefighters, JSDH, .NET, Android, HTTP, WebSocket, Web aplication, Web push notification

VOPÁLKA, Matěj. *Aplikace pro dobrovolné hasiče*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 55 s. Bakalářská práce. Vedoucí práce: Ing. Tomáš Gerlich,

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Matěj Vopálka
VUT ID autora:	220834
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Aplikace pro dobrovolné hasiče

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Tomášovi Gerlichovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Systém svolávání dobrovolných hasičů k zásahu aktuálně postrádá efektivní indikaci počtu účastnících se hasičů. Dobrovolní hasiči se svolávají k výjezdům různými druhy, jako jsou zprávy SMS, nebo sirény. Je nutné vědět, kolik členů jednotky se daného výjezdu účastní. To vede k hledání alternativních řešení jako jsou telefonní hovory, nebo zprávy na sociálních sítích. Pro hasičský zásah je kritická rychlost a počet účastnících se hasičů, proto ani telefonní hovor, ani zprávy nejsou vhodnou indikací.

Z tohoto důvodu bude vyvinuta multiplatformní aplikace pro dobrovolné hasiče plnící roli oznámení nového výjezdu, potvrzení výjezdu hasičem a indikaci tohoto stavu.

V bakalářské práci bude čtenář seznámen se základy pro vybrání vhodného jazyku při vývoji. Dále se práce zabývá popisem základních protokolů pro internetovou komunikaci, vývojem aplikace ve frameworku ASP .NET core a vývojem aplikace pro Raspberry Pi. Dále popisuje nasazení aplikace jako služby operačního systému Raspbian a nasazením webové aplikace na cloud Azure.

Poslední kapitoly jsou věnovány popisu architektury aplikace, možného napojení na již existující a nové řešení a seznámením s uživatelským rozhraní aplikace.

1 Návrh řešení

Pro případy nebezpečí člověka jsou zřízeny tísňové linky. Existují hlavní čtyři tísňové linky, pod čísly 150, 158, 155 a 112. V České republice jsou nepostradatelnou součástí integrovaného záchranného systému jednotky sborů dobrovolných hasičů, kteří dobrovolně zasahují při krizových událostech a ulehčují tak práci Hasičskému záchrannému sboru České republiky.

Dobrovolné jednotky jsou svolávány pomocí většího množství technologických řešení. Práce se bude zaměřovat na řešení pomoci síťových aplikací. Tyto aplikace budou mít různé architektury, kde každá má své výhody a nevýhody. Typy architektur budou v této kapitole probrány a zhodnoceny.

1.1 Aktuální stav

Pro správné porozumění architekturám je potřeba nahlédnout do aktuálního stavu řešení svolávání dobrovolných jednotek požární ochrany.

V současném stavu v případě, kdy člověk v nesnázi zavolá na tísňovou linku 150, nebo 112 je spojen s operátorem Krajského operačního a informačního střediska (KOPIS). Následně operátor zjišťuje od volajícího stav, aby zjistil, jestli je opravdu nutno někam poslat jednotky (například se na hasičský záchranný sbor dají nahlásit plánovaná pálení a při nahlášení požáru v této lokalitě operátor volá na číslo nahlašovatele pálení, aby zjistil, jaký je stav), kolik jich má poslat (zde je nutné aby, operátor odhadl, jak je situace vážná, protože vyděšený člověk může popsat situaci jinak, než je skutečný stav a poté mohou jednotky chybět na místě zásahu, nebo naopak na stanici, pokud by se v čase zásahu udála další událost.) a kam jednotky poslat (ačkoliv se tento úkol může zdát snadným, je občas těžké z popisu oznamovatele zjistit, kde se daná událost odehrává a je žádoucí a důležité, aby jednotky dorazili na místo co nejdříve a zabránily tak co nejvíce škodám). Příklady výše zmíněných problémů jsou tyto tři události:

1. Výjezd k zásuvce ve které zajiskřilo. Jednotka dobrovolných i profesionálních hasičů a jeden vůz policie jeli k události, u které zajiskřilo v zásuvce. Majitel vypnul hlavní elektrický spínač a vyčkal příjezdu jednotek požární ochrany.
2. Výjezd k požáru domova důchodců. Tato událost byla nahlášena jako požár domova důchodců a vyjely k ní čtyři jednotky požární ochrany. Po příjezdu první jednotky bylo průzkumem zjištěno, že se požár týká chaty poblíž domova důchodců a tak přebytečné jednotky opustili místo zásahu a jeli zpět na stanici.
3. Požár u hospody s fotbalovým hřištěm. Výjezd byl nahlášen jako požár u hospody s fotbalovým hřištěm. Po příjezdu na místo bylo průzkumem zjištěno, že

v okolí žádný požár nehrozí a po telefonickém dotázání o upřesnění oznamovatele bylo zjištěno, že požár je u jiné hospody ve stejné městské části, která má vedle sebe také fotbalové hřiště.

Poté, co operátor KOPIS zjistí stav události, vyšle podle vážnosti jednotky požární ochrany. Pro dobrovolné jednotky se potom liší druh svolávání. Některé jednotky, převážně na vesnicích, se svolávají sirénou. Pro více osídlené oblasti s vyšším počtem událostí tento způsob není přívětivý pro obyvatelstvo a tak se používají jiné způsoby. Jedním z nich je výjezdová SMS a výjezdové prozvánění.

1.2 Problémy aktuálních řešení

Způsobů jak svolávat hasiče je mnoho a každé má své výhody a nevýhody. Níže budou probrány problémy aktuálních řešení svolávání dobrovolných hasičů.

- Nevýhoda SMS výjezdu je v tom, že na některých telefonech se nedá nastavit vyzvánění pro SMS, nebo například v tichém režimu nejdou slyšet vůbec.
- Předchozí nevýhodu kompenzuje nedávno zavedené výjezdové prozvánění, při kterém se již na většině telefonů dá nastavit speciální vyzvánění a na některých lze i nastavit vyzvánění i při tichém režimu. Obě výše zmíněné varianty však mají problém při řešení zpětné vazby od člena výjezdové jednotky, kde si každá jednotka řeší potvrzení výjezdu po svém.
- Některé jednotky potvrzení nemusí řešit vůbec a zjistí stav až po příchodu na hasičskou zbrojnici. Ovšem takové řešení může například způsobit menší počet členů zapojených do výjezdu, nebo pozdější zjištění o nemožnosti výjezdu jednotky z důvodu nedostatečného počtu členů.
- Některé jednotky například píší o své účastni po chatu na sociálních sítích, jako je například Facebook. Takovéto řešení umožňuje větší účast na výjezdech, jelikož ostatní členové vidí, kolik členů se přihlásilo a mohou chvíli počkat na opožděné členy, ovšem pouze v rámci doby, do které musí vyjet (např 10 minut pro třetí stupeň jednotky požární ochrany), nebo dřívější zjištění o nedostatečném počtu členů, nebo-li nemožnosti výjezdu (jednotka není akceschopná). Toto řešení ale není optimální, jelikož například při události typu požár není času nazbyt a je potřeba vyjet co nejrychleji a odepisování ve spěchu je poměrně náročné. Někteří členové by tak museli dávat vědět o potvrzení výjezdu za volantem osobního automobilu, při běhu, řízení jízdního kola, či jiného dopravního prostředku. Navíc i při nepřijetí výjezdu je potřeba tuto skutečnost oznámit, aby jednotka zbytečně nemusela čekat na členy, a tak by někteří členové museli odepisovat například při práci na střeše nebo jiných nepřívětivých okolnostech. Nevýhodou taky je fakt, že ne každý člen si vzpomene otevřít

chatovou aplikaci před výjezdem jednotky, a tak i přes oznámení členů o jejich akceschopnosti mohou vyjet bez nich a zmenšit tak jejich počet.
- Další nevýhoda by mohla být nepřítomnost mobilního zařízení například při práci na stolním počítači, kdy by člen jednotky nemusel mobilní telefon slyšet, například z důvodu nabíjení baterie, nebo jejího úplného vybití, přitom se nabízí možnost upozornění přímo na počítači.

1.3 Řešení aktuálních problémů

Jako řešení výše zmíněných problémů by mohla být mobilní aplikace, která by naslouchala pro příchozí události, upozornila a dotázala se člena jednotky na jeho akceschopnost, kterou by následně oznámila serveru, který by zobrazil informace o akceschopnosti členů na obrazovce / obrazovkách umístěných mezi hasičské automobily tak, aby na ně strojníci dobře viděli. Toto řešení by tak pomohlo zvětšit množství členů jednotky na výjezdech, zrychlit potvrzení o účasti na výjezdech a zmenšit riziko z nepozornosti člena jednotky při výjezdu.

Zároveň se zde nabízí možnost monitorování statistik pro lepší přehlednost doby, kdy je jednotka nejvíce akceschopná, analyzování o počtu výjezdů v závislosti na jevech jako je teplota, rychlost větru, vlhkosti vzduchu, dne v týdnu a dalších. Pro takovou analýzu by bylo nejvhodnější nasadit umělou inteligenci, avšak její trénování potřebuje velké množství dat. Existují i další, již funkční řešení, jako je například aplikace výjezdová SMS, která čte SMS zprávy a umožňuje nastavit jim vyzvánění i přes tichý režim a následné přeposílání, nebo odpověď, nebo komplexní komerční řešení Fireport.

1.4 Návrhy řešení

Z pohledu této práce budou probrána následující řešení:
- PWA aplikace
- Nativní aplikace
- Nativní aplikace bez čtení SMS
- PWA aplikace bez čtení SMS

1.4.1 PWA aplikace

Webový a API server s použitím progresivních webových aplikací. Toto řešení by bylo jednoduché z hlediska multiplatformnosti. Z hlediska přístupnosti osobních počítačů, ať už z hlediska mobilních telefonů, osobních i pracovních laptopů a stolních počítačů, skoro všechny disponují webovým prohlížečem, a proto se nabízí vytvořit

pouze jednu webovou aplikaci, která by jak četla příchozí SMS, tak umožňovala zobrazení členů, kteří potvrdili výjezd. Zároveň se zde nabízí možnost pro naslouchání výjezdů nejen z SMS, ale i pomocí internetové komunikace. Navíc lze v dnešní době progresivní webové aplikace nainstalovat na zařízení, a tak se pro uživatele stávají velmi podobnými nativním. Progresivní webová aplikace se otevře v prohlížeči ve speciálním celoobrazovém módu, takže ani není vidět okno pro zadání URL adresy. Bohužel toto řešení není jednoduché provést a to z důvodu nepřítomnosti API pro čtení SMS z bezpečnostních důvodů. Existuje pouze API pro autorizační SMS, které v tomto případě nelze použít. Navíc není jednoduché upozorňovat uživatele na výjezd pomocí notifikací z hlediska webové aplikace. [1] [2]

1.4.2 Nativní aplikace

Vzhledem k nelehké implementaci výše zmíněného řešení se nabízí možnost nativních aplikací. Nativní aplikace jsou aplikace vytvořené přímo pro konkrétní systém. Vzhledem k převaze mobilních telefonů na trhu firmy Apple a telefonů se systémem Android se nabízí zvolit nativní aplikace právě pro tyto dva druhy zařízení.

Pro mobilní telefony firmy Apple to znamená vytvořit aplikaci pro operační systém iOS a pro operační systém Android zvolit Android studio. Zatímco vývoj pro Android není až tak náročný na druh, zařízení u vývoje na zařízení se systémem iOS nastává problém z hlediska vývojového prostředí. Není totiž možné vyvíjet aplikace bez telefonu, či jiného zařízení s iOS od firmy Apple, nebo jejich počítači se systémem MacOS. Pro Android je vývoj méně náročný, je totiž nutné, buď stejně jako u iOS mít zařízení se systémem Android, a nebo ho jednoduše emulovat přímo ve vývojovém prostředí. U firmy Apple je emulace možná právě na počítačích se systémem MacOS.

Další možností pro vývoj nativních aplikací je například platforma Xamarin, která umožňuje vývoj zároveň pro zařízení s Android i iOS. Xamarin má svoji syntax pro tvoření grafického uživatelského rozhraní a využívá programovací jazyk C#, stejně jako další jazyky z rodiny dotnet, jako například F# a Visual Basic.

Existují i další možnosti pro vývoj na obou zařízení zároveň, a však je zde opět problém přístupem k SMS zprávám a to z hlediska omezenosti iOS, který neumožňuje číst SMS zprávy z hlediska bezpečnosti jejich uživatelů. Potencionální útočník by tak mohl vytvořit aplikaci, která bude špehovat uživatele zařízení a přeposílat jeho SMS zprávy. Na zařízení se systémem Android sice lze SMS zprávy číst, a však se zvyšujícími se verzemi je tuto funkcionalitu těžší implementovat a nebo například musí být aplikace, která chce SMS zprávy číst nastavena jako defaultní aplikace pro čtení SMS zpráv, což by znamenalo omezení uživatele ve volbě jeho aplikace při použití s tou pro oznámení o výjezdu a implementace veškerých funkcionalit SMS ve vyvíjené aplikaci.

1.4.3 Nativní aplikace bez čtení SMS

Kvůli výše zmíněným problémům je vhodné uvažovat nad převrácením celého konceptu a nepřijímat SMS zprávy na zařízeních, ale na serverové části. Pro tuto funkcionalitu je nutné použít GSM modul, který bude komunikovat s mikrokontrolérem, či počítačem. V tomto případě by webový server dostával upozornění o výjezdu například pomocí webového API (Application Programming Interface) a notifikoval připojené klienty přes internetovou síť o přijaté SMS zprávě o výjezdu. Tato možnost je i velice výhodná z pohledu rozšířitelnosti, protože se dá vyměnit SMS komunikace rovnou za internetovou, která je daleko bezpečnější možností. SMS zprávy totiž lze rozesílat falešně z vybraného čísla a tak by potencionální útočník mohl rozesílat falešné výjezdy a omezovat tak akceschopnost dobrovolných jednotek požární ochrany.

Toto řešení má však nevýhodu ve skutečnosti, že stále velké množství lidí, nemá stálé připojení internetu pomocí mobilního operátora. Z tohoto důvodu se nabízí varianta ponechání původního řešení namísto jeho nahrazení a přidat k němu nový systém. Přidání k aktuálnímu systému sice znamená zachování rizika falešných SMS a dokonce dvojí upozornění, avšak lze ji vyměnit, když bude potřeba. Při nahrazení novou aplikací s internetovým rozesíláním by jednotky mohli utrpět v počtu členů, kteří stále využívají obyčejné tlačítkové telefony.

1.4.4 PWA aplikace bez čtení SMS

Kvůli problémům se čtením SMS v telefonech a alternativním řešení v podobě čtení SMS zpráv na počítači odpadá výše zmíněný problém PWA aplikací, a tak se nabízí jejich použití pro jejich multiplatformnost. Progresivní webové aplikace se chovají jako stránka v prohlížeči a podle manifest souboru přibaleného a definovaného v úvodní stránce lze upravit vzhled prohlížeče do různých módů, například bez řádku pro zadávání URL adresy. [3]

Obr. 1.1: Schéma PWA aplikace

Základ aplikace je JavaScriptový soubor nazývaný service worker, který se při PWA aplikaci chová jako proxy server. Schéma PWA aplikace je zobrazeno na obrázku 1.1. Aplikace zaznamenává jednotlivé dotazy na webový server, ale nepředává je rovnou webovému serveru, ale service workerovi, který je sám zpracovává a lze

tak například využít úložiště v prohlížeči pro ukládání neměnných stránek, nebo pro nakešování dynamických, pro zvýšení rychlosti. [4]

Schopnosti PWA se liší dle podpory daného prohlížeče a ty se v dnešní době značně liší, nicméně podpora stále roste. Některé prohlížeče umožňují zasílat i tak zvané Push notifikace. V případě PWA se jedná konkrétně o Webové push notifikace. Bohužel k dnešnímu dni ještě stále nejsou Webové push notifikace podporovány na zařízeních s iOS firmy Apple, nicméně na začátku roku přibyla experimentální možnost tyto notifikace zapnout, avšak v další aktualizaci systému opět chybí. Je tedy pravděpodobné, že se firma Apple bude snažit o jejich podporu. [5]

2 Vývoj aplikace

V rámci práce byla zvolena možnost progresivní webové aplikace bez čtení SMS. To znamená implementaci webového serveru s webovým aplikačním rozhraním a responzivním uživatelským rozhraním. Zároveň implementaci zařízení pro čtení SMS zpráv pomocí GSM modulu, které bude realizovat přeposílání HTTP požadavků webovému serveru. Pro implementaci webového serveru je třeba zvolit vhodný programovací jazyk, porozumět protokolu HTTP a vybrat vhodný typ databáze.

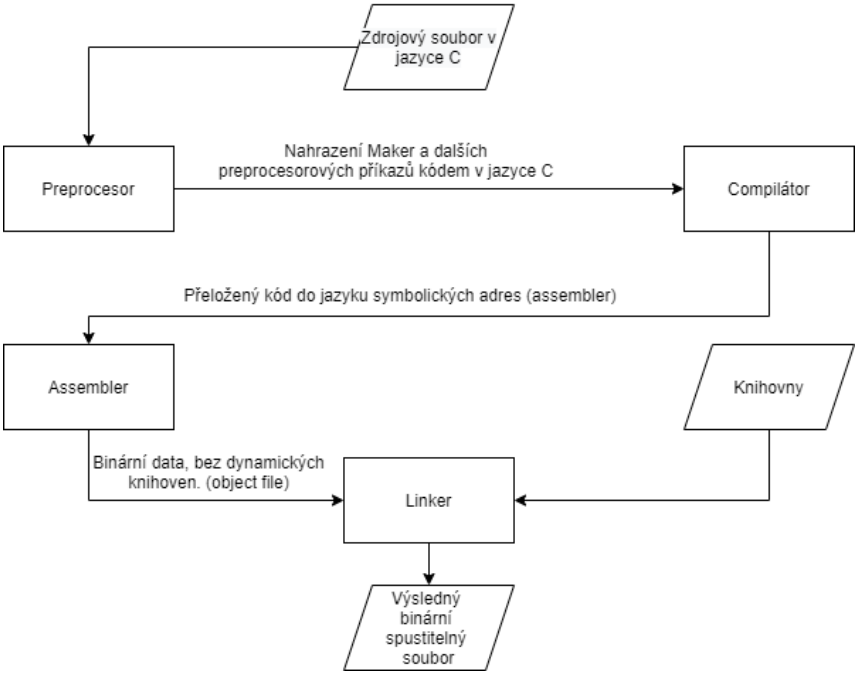
2.1 Typy jazyků

Programovací jazyky od svého vzniku prošly několika velkými změnami. Původní programování se realizovalo pomocí binární formy, kde každá instrukce má své jedinečné binární číslo. Tyto čísla se pak ukládali do paměti programu, kterou pak procesor prochází pomocí čítače instrukcí. Vzhledem k tomu, že při větší projektech se člověk na rozdíl od počítače vyzná v programu těžce, vznikly první programovací jazyky, mezi které například patří Assembler.

Assembler usnadňuje vývoj překládáním textově psaných instrukcí na čísla a tím umožňuje programátorovi psát kód přehledněji, než tomu bylo u binární formy. Nevýhodou je fakt, že pro každý typ procesoru existuje jiná instrukční sada a tak se napsaný kód nedá spustit na jiných typech. V assembleru pak programátoři opakovaly určité postupy, jako jsou například cykly a pro neustále opakování vznikly jazyky vyšší úrovně, které již měli pro cykly jednoduchý zápis.

Tyto jazyky je však potřeba do assembleru přeložit a proto se používají takzvané kompilátory, které jazyky jako je například jazyk C přeloží do assembleru. Proces kompilace je zobrazen na obrázku 2.1. Kompilování jazyků přináší další výhodu a to možnost multiplatformnosti. Pro každý procesor, na který byl naprogramován kompilátor lze zdrojový kód C zkompilovat na instrukce právě pro jeho typ. Protože však jazyk C operuje pořád na dost nízké úrovni a byly vyvíjeny další způsoby programování, vznikl například jazyk Python. [6]

Jazyk Python pracuje na odlišném způsobu než jazyk C. Python je sám o sobě program, ve kterém se spouští programy napsaném ve stejnojmenném jazyce. Kompilace do binární podoby programu se děje za běhu, a tak lze v Pythonu i psát kód za běhu. Python podporuje strukturované programování, funkcionální programování i objektově orientované. Nevýhodou Pythonu může být nižší rychlost běhu, kvůli kompilaci za běhu, neboli nekompiluje se, nýbrž interpretuje. Jeho výhodou je naopak větší abstrakce, velké množství knihoven. Dalším způsobem vykonávání kódů je kombinace obou výše zmíněných způsobů dohromady, kterou využívají například programovací jazyky Java a C#.



Obr. 2.1: Proces kompilace jazyka C

Tyto jazyky se přikládají vždy prvně do speciálního strojového kódu, který vstupuje do virtuálního stroje. V případě Javy to je Java virtual machine, nebo pro C# .NET runtime. Tento strojový kód se následně interpretuje přímo do skutečného strojového kódu, kterému rozumí procesor. Virtuální stroj tedy funguje jako interpreter, do kterého vstupuje již připravený zkompilovaný kód a tak tyto jazyky kombinují jak výhody interpreteru, tak rychlost kompilace, avšak vzhledem k vyšší úrovni abstrakce jsou tyto jazyky pomalejší a jejich celková velikost (programu i virtuálního stroje) je vyšší.

2.2 Protokol HTTP

Protokol HTTP je bezstavový protokol operující na aplikační, neboli sedmé vrstvě referenčního modelu ISO/OSI, nebo také čtvrté vrstvě modelu TCP/IP. Protokol HTTP je defaultně přístupný na TCP portu 80, avšak existují i varianty s UDP a číslo portu lze změnit. Protokol je používán k přenosu souborů jako jsou HTML stránky, kaskádové styly, JavaScript, multimediální soubory, aplikační data a ostatní. Skládá se z metody která může nabývat nejčastěji hodnot GET, POST, PUT a DELETE. Existují však i další metody. Po metodě následuje cesta k tázanému souboru a verze protokolu HTTP. Poté následuje konec řádku a hlavičky, což jsou další rozšiřující data. Po hlavičkách následují dva konce řádku a samotná data souboru. [7] [8]

2.2.1 Bezpečnost

Protože tvůrci původního protokolu HTTP nepředpovídali směr budoucího vývoje internetu, neuvažovali nad bezpečností tohoto protokolu a tak není šifrovaný, proto později vznikl protokol HTTPS který je postavený na principu HTTP s použitím protokolu SSL/TLS.

2.2.2 Metoda GET

Metoda GET nejčastěji slouží pro získávání dat ze serveru. Její parametry se píší přímo do URL adresy a tak není příliš vhodná pro zasílání hesel a ostatních údajů. Daleko větší využití má pak v případě, kdy chceme na určité adresy vždy obdržet stejná data, jak je například v případě e-shopů, uživatelských účtu a podobných dat.

2.2.3 Metoda POST

Metoda POST na rozdíl od metody GET neposílá parametry v URL adrese, a proto je daleko vhodnější pro formuláře a manipulaci s daty. Metodu GET je například vhodné použít při registraci uživatelů, produktů, či jiných databázových údajů, stejně jako například pro přihlašování uživatelů do systému.

2.3 Webový server

Webový server je počítač, který naslouchá a odpovídá na HTTP dotazy. Webových serverů existuje celá řada, mezi hlavní představitele pak patří Apache, Ngnix, nebo IIS. Pro framework ASP .NET core, který slouží pro vývoj webových aplikací za použití C#, je pak zajímavý webový server Kestrel, který je přímo součástí frameworku a je tak možné naprogramovanou aplikaci přímo spustit. U webového serveru je důležitá rychlost obsluhování HTTP požadavků, rychlé připojení k síti a stabilita. Existují různé typy webových stránek, mezi základní patří:

1. Statické stránky
2. Dynamické stránky
3. Interaktivní stránky

Pro komplexnější aplikace lze použít na stránkách i technologii webových socketů a webových push notifikací.

2.3.1 Statické stránky

Dřívějším nedostatkem byly pouze statické webové stránky, realizované dokumenty psanými ve značkovacím jazyce HTML (Hypertext Markup Language). Tyto stránky se zobrazovali v prohlížeči bez možnosti jiné interakce s uživatelem, než-li přejít na jiný HTML dokument. [9]

2.3.2 Dynamické stránky

V průběhu let však byla potřeba z hlediska různých e-shopů tyto stránky generovat, a tak se začal uplatňovat princip dynamicky generovaných HTML dokumentů. Tyto dokumenty se generují při HTTP požadavku od HTTP klienta, kterým může být jak webový prohlížeč, tak jiný program jako například curl, nebo JavaScript. Při dotazu klienta webový server dle požadavků vygeneruje za pomocí HTML šablon a programovacího jazyka (většinou s pomocí frameworku) HTML stránku. Využívá se tak například při realizaci e-shopů, kde klient zašle požadavek na zboží pomocí metody GET a server nalezne informace o daném zboží v databázi a vrátí stránku dle šablony s přidanými daty o zboží. Díky tomuto principu se šetří velké množství místa a čas, ve kterém by se musely všechny stránky vytvářet ručně. [10]

2.3.3 Interaktivní stránky

Některé webové aplikace trpí kvůli základnímu bezstavovém konceptu HTTP a potřebují aktualizovat svůj obsah. Právě k tomuto účelu slouží programovací jazyk Javasript, který umožňuje vykonávat kód přímo v prohlížeči klienta. Právě díky Javasriptu je možné vytvářet například filtry dat, nebo možnost aktualizovat pouze část stránky a nepřenášet tak data, která již byla přenesena a tak zrychlit načítání.[11]

2.3.4 Webové sockety

I přes výše zmíněné nastává stále problém pro aplikace, které musí reagovat na událost co nejdříve po obdržení. Pro tyto aplikace není vhodné neustále aktualizovat data ze serveru opakovaným dotazováním, které by vedlo spíše k simulaci útoku DoS a server by tak mohl přestávat stíhat všechny dotazy zpracovávat. Namísto neustálého dotazování se serveru je vhodné využít webové sockety. Tyto webové sockety se snaží nahradit nevýhodu bezstavového protokolu HTTP tím, že na spojení neodpovídají okamžitě, ale drží spojení otevřené, a tak může nejen HTTP klient notifikovat server, ale i naopak server, notifikovat HTTP klienta.[12]

2.3.5 Webové push notifikace

Webové sockety jsou výbornou formou pro zasílání aktuálních dat, ale v případě kdy je aplikace na pozadí operační systémy z důvodu úspory výkonu a baterie, kód na pozadí nenechávají běžet. V případě aplikace, která musí uživatele notifikovat o novém stavu je velký problém schopnost aplikace notifikovat jen při otevřeném stavu. Pro tuto funkcionalitu existují Webové push notifikace. Webové push notifikace jsou zabudovány přímo v prohlížeči, a tak programátor nemusí řešit implementaci logiky pro komunikaci, čekání na notifikaci, ani problémy s uspáváním aplikace operačním systémem.

2.4 Databázové systémy

Pro ukládání dat mohou při malém rozsahu posloužit soubory na souborovém systému operačního systému, avšak pro větší množství dat tato možnost není vhodná jelikož se stává nepřehlednou a pomalou. Z tohoto důvodu se využívají databáze, které umožňují mnohem rychlejší přístup k potřebným datům, přístup více uživatelů k datům zároveň a usnadňují práci s daty. V dnešní době se hlavně využívají databáze relační a objektové.

2.4.1 Relační databáze

Tyto databáze jsou striktně definované a mají podobu tabulek podobných jako známe z tabulkových procesorů jako je Excel, Numbers, nebo Calc. Struktura dat v těchto databázích se těžko mění, a tak je důležité ji dobře definovat na začátku při vytváření databáze. Pro komunikaci s těmito databázemi se využívá jazyka SQL, pomocí kterého se databáze vytváří, vkládají se do ni data, dotazuje se na tyto data a upravují se tyto data.

2.4.2 Objektové databáze

Objektové databáze spíše připomínají princip objektově orientovaného programování, a tak se s nimi dá dobře pracovat. V aktuální době však slouží spíše jako rozšíření pro databáze relační. Vzhledem k jejich objektové struktuře se dá daleko lépe upravovat struktura dat.

2.4.3 Definice dat

V databázích mohou být data různě strukturovaná dle potřeby. V relační databázi mohou mít různý počet sloupců, nebo vazeb. Vazby se využívají pro spojení více

záznamů k sobě na základě vztahu. Druhy vazeb:

1. Bez vazby - Data nenavazují na žádná jiná.
2. Vazba 1 na 1 - Tato vazba je velmi jednoduchá, je třeba pouze vytvořit v jedné tabulce pole s identifikačním číslem záznamu v tabulce druhé a popřípadě naopak.
3. Vazba 1 na "n" - Tato vazba se řeší většinou jednosměrně a to z tabulky, která obsahuje více prvků odkazující na jeden. Princip je stejný jako u vazby 1 na 1. V tabulce obsahující data "n" přidáme pole, které odkazuje identifikačním číslem na záznam v tabulce obsahující data "1". Více dat v tabulce "n" může odkazovat na tentýž záznam v tabulce "1"
4. Vazba "m" na "n" - Tato vazba je nejsložitější a musí se ve většině případů převádět na 2 vazby 1 na "n". Vazba se řeší spojovací tabulkou, která obsahuje spojovací záznamy, které obsahují identifikační číslo prvku z tabulky "n" i identifikační číslo prvku z tabulky "m".

2.5 Operační systém Android

Android je otevřený operační systém, založený na linuxovém jádru. Je to otevřený projekt, který primárně vyvíjí firma Google společně s Open Handset aliancí. Tento systém byl vydán roku 2008 a naprogramován v programovacím jazyce Java, přičemž jeho jádro je napsáno v jazyku C a C++.

Pro případ webové progresivní aplikace se o její chod stará prohlížeč, takže není třeba v takové míře řešit samotný operační systém. Různé prohlížeče podporují různé funkce PWA aplikací. Například skoro všechny moderní prohlížeče podporují funkci webových push notifikací, kromě systému iOS, který ve svém prohlížeči safari začlenil experimentální možnost těchto notifikací v lednu 2022. Toto nastavení Apple později pozměnil a budoucí vývoj není v současné době znám. Ostatní systému umožňují prohlížeče s podporou webových push notifikací.

2.6 Raspberry Pi

Raspberry Pi je plnohodnotný jednodeskový počítač, založený na architektuře ARM o velikosti platební karty. Mimo standardní konektory porty, jako je Ethernet, USB a HDMI disponuje i GPIO piny, ke kterým je možné připojit další zařízení, jako senzory, různé moduly i další počítače. Má velkou výhodu ve své velikosti a malé spotřebě. Navíc má na rozdíl od své konkurence daleko větší podporu, i když ta zase disponuje zajímavějšími možnostmi konektivity, jako je například konektor M2 pro připojení NVMe SSD.

Raspberry Pi pro svůj běh potřebuje operační systém. Těch je celá řada, přičemž výrobci sami vyvíjejí operační systém založený na linuxovém jádře Raspbian. Raspberry Pi nemusí sloužit pouze pro notifikování webového serveru o výjezdech z SMS, ale může také posloužit jako zobrazovací zařízení pro výjezdy po připojení monitoru, nebo jako webový server.

V případě použití Raspberry Pi pro notifikování webového serveru o SMS je třeba GSM modul, který lze připojit například k sériové lince Raspberry Pi. Jedním takovým modulem je například GSM modul SIM800L, komunikující přes AT příkazy.

3 .NET

.NET je platforma pro vyvíjení aplikací v jazycích C#, F#, Visual Basic, iron Python a dalších. Všechny aplikace .NET pracují na virtuálním strojem .NET runtime, který musí být podporován pro platformu, na které má kód pracovat.

Pro .NET je k dispozici několik frameworků, pro vývoj webových, mobilních, i multiplatformních aplikací, her a další typy aplikací.

3.1 Vývoj aplikace v ASP .NET core

.NET nabízí více uživatelských rozhraní pro ovládání. Mezi ně patří například Visual Studio, ostatní vývojová prostředí, jako jen příklad Rider, nebo terminálové rozhraní .NET CLI, které umožňuje vytvářet projekty rovnou z příkazové řádky. [13]

3.1.1 Šablony

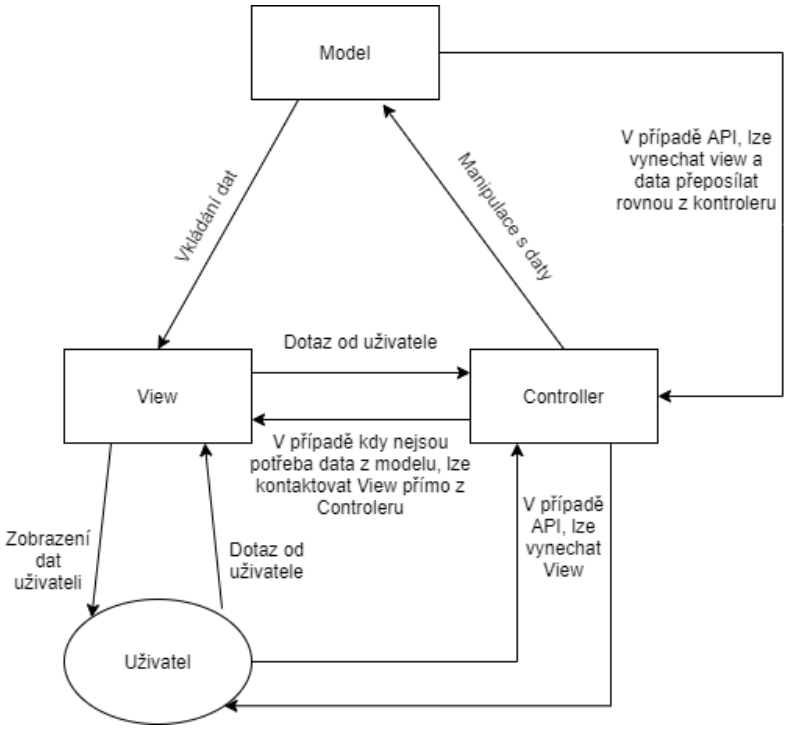
Protože je vývoj webových aplikací velice rozšířený a v dnešní době převažuje, je v .NET rovnou několik již předpřipravených šablon. Tyto šablony umožňují rychlejší začátek vývoje a jednodušší ovládání pro začátečníky.

Pro velký počet, zde bude uvedeno jen pár příkladů šablon

1. console - Tato šablona vygeneruje pouze jeden soubor pro zdrojový kód a jeden adresář pro konfigurační soubory.
2. sln - Tato šablona generuje solutionfile, který zpravuje jednotlivé projekty. Projekty mohou být úplně na odlišných místech v souborovém systému operačního systému, ale při otevření .sln souboru se otevřou všechny projekty, které solutionfile zpravuje. Lze tak například používat pouze jednou vytvořené modely v několika aplikacích například webová aplikace, webové API, mobilní aplikace, konzolová a další.
3. webapp - Tato šablona vytváří přímo dynamické webové stránky za pomocí razorpages, což je jazyk příbuzný HTML, který podporuje i zápis C# kódu. Tyto Razor pages pak generují HTML soubor, který se posílá klientovi.
4. webapi - Tato šablona generuje webové API, které rovnou obsahuje základní strukturu pro získávání dat o počasí, která jsou náhodně generována.

Návrhový vzor MVC

Návrhový vzor MVC se skládá ze tří hlavních komponent a to Model, View, Controller. Jeho principem je zamezení takzvaného špagetového kódu, který se vyznačuje vysokou nepřehledností kódu. Návrhový vzor MVC je zobrazen na obrázku 3.1. V případě špagetového kódu, bychom například mohli mít jeden velký soubor, ve



Obr. 3.1: MVC

kterém by bylo několik metod, vracející HTML soubory a bylo by tak velice obtížné měnit návratové hodnoty funkcí.[14] [15]

Návrhový vzor MVC tomuto zabraňuje tím, že kód dělíme na jednotlivé části podle toho, co ve skutečnosti dělají. [16]

- Model - Stará se o strukturu dat aplikace. Druh zpracování se může lišit dle typu a použití dat. Například může model generovat náhodná data, spravovat pouze data v rámci daného spuštění aplikace, pracovat s trvalými daty ve formě souborů, nebo přistupovat k databázi s daty. Pro práci s daty se využívá controller, popřípadě se dá využit nástroj pro přímou manipulaci s daty jako textový editor pro přepsání dat v souboru, nebo program pro management databáze, jako je například nástroj SQL Server Management Studio (SSMS) od Microsoftu.
- View - Stará se o vzhled a strukturu uživatelského rozhraní. Je to ta část systému, se kterou uživatel interaguje. Zobrazuje mu data z aplikace, která dostává z modelu a naopak je uživatel pomocí view vkládá do controlleru, který je zpracovává a předává modelu. Často využívá speciální jazyky, které slučují jazyk pro definici grafického rozhraní, jako HTML, nebo XML s programovým kódem jako je C#, Java, Python a další pro zpřístupnění dat. Příkladem takového jazyka je Razor pages, nebo rozšíření webového frameworku Spring, pro jazyk Java Thymeleaf.

- Controller - Controller se stará o samotné dotazy. Naslouchá na určených URL adresách pro příchozí dotazy a zpracovává je. V případě příchodu nových dat, nebo úpravě stávajících kontaktuje model pro jejich aktualizaci a následně může poslat data z modelu do view, které se postará o jejich strukturu a vzhled. V případě kdy uživatel nežádá žádná dynamická data, může controller kontaktovat rovnou view a zobrazit pouze statickou stránku. Naopak pokud chce uživatel pouze manipulovat s daty, lze vynechat view a controller přistoupí pouze k modelu a vrátí data v podobě XML, JSON, YAML, nebo další. Dále v controlleru může být autentizace.

Adresářová struktura

Do adresáře Controller se vkládají jednotlivé controllery tak, jako je uložen defaultní controller WeatherForecastController.cs. Firios.API.scproj je pak soubor spravující závislosti na ostatních projektech, nebo balíčkách nuget, které jsou spravovány stejnojmenným balíčkovým manažerem. Program.cs je hlavní programový soubor, který se spouští na začátku a je v něm uložena konfigurace webového serveru. Ve složce Properties jsou uloženy soubory pro konfiguraci, například soubor launchSettings.json, kde se například nachází konfigurace IP adres serveru pro HTTP, HTTPS a jejich porty. WeatherForecast.cs je model pro data o počasí, správně by tento model měl být uložen v adresáři Models spolu s ostatními modely. Soubory appsettings jsou pro další konfiguraci, například pro konfiguraci logování. Složka obj slouží při kompilaci jako mezikrok.

.Net šablona lze v základě spustit a obsahuje soubory pro příklady tříd, které jsou ve většině aplikací zbytečné a tak mohou být klidně odstraněny. Jedná se tedy například o třídu controller WeatherForecastController.cs a model WeatherForecast.cs

Controllery

Visual studio umožňuje automatické vygenerování controlleru podle šablony, což ulehčuje mnoho práce, avšak výsledek této operace umožňuje jen základní dotazy a tak je potřeba ji upravit. Šablony umožňují vytvořit mimo jiné dva druhy controllerů.

1. API Controller - který při komunikaci vynechává view a odesílá pouze čistá data ve formě textu, dat ve formě souboru JSON, XML, či jiných souborů.
2. Controller pracující s view - který při na svém konci provolává view, popřípadě do něj posílá data, která se vkládají do šablony.

Controller ovládající API bude sloužit pouze k přenosu dat a to primárně ve formátu JSON. Bude obsahovat metodu pro zobrazení dat k výjezdu dle ID výjezdu,

registraci nového výjezdu, registraci uživatele k výjezdu, registraci webového socketu a registraci dat k push notifikacím.

Ostatní controlery budou ovládat části webu dle svého logického zařazení. Home controler bude obsluhovat základní část webu, Incident controller bude obsluhovat stránky zabývajícími se výjezdy a user controller bude obstarávat uživatelská data.

3.2 Entity Framework

Kvůli jednodušší manipulaci s databází mají různé jazyky různé Objektově relační mapování (ORM). Tyto ORM jsou obvykle dostupné ve formě balíčků a umožňují pracovat s databází na úrovni objektu programovacího jazyka bez použití SQL příkazů. Pro .NET je nejpoužívanější ORM EntityFramework. Pro jeho implementaci je potřeba i dalších balíčků, kvůli možnosti připojení k databázi. Nuget řeší závislosti balíčků mezi sebou, a tak není potřeba hledat a instalovat jednotlivé balíčky, avšak v případě kdyby bylo třeba použít dvě různé databáze, bylo by třeba nainstalovat balíčky pro obě tyto databáze. [17]

Pro vývoj aplikace byla zvolena databáze Microsoft SQL server, pro kterou má Entity Framework připraven balíček Microsoft.EntityFrameworkCore.SqlServer. Pro práci s databází je vhodný i balíček Microsoft.EntityFrameworkCore.Tools, který umožňuje generování takzvaných migrací. Balíčky lze pomocí nuget instalovat více způsoby. [18]

1. Pomocí grafického uživatelského rozhraní ve Visual Studio, nebo jiných vývojových prostředí. Lze jednoduše vyhledávat a vybírat verze pomocí grafických prvků
2. Pomocí PackageReference - Píše se rovnou do souboru .csproj do XML elementu ItemGroup. Musí se specifikovat přesný název balíčku a jeho verze.
3. Pomocí dotnet CLI - terminálová aplikace vloží název balíčku a jeho verzi do .csproj

Pro nainstalování balíčků stačí napsat:

```
dotnet add package Microsoft.Microsoft.
   EntityFrameworkCore.SqlServer --version 6.0.0
dotnet add package Microsoft.EntityFrameworkCore.Tools --
   version 6.0.0
```

Další důležité rozšíření pro práci s Entity Frameworkem je terminálová aplikace, která pomáhá vytvářet migrace, aktualizovat a vytvářet databázi. Pro aplikace vázané na vývoj daného projektu je dobrým zvykem používat stejné verze, aby mezi jednotlivými vývojáři nevznikali konflikty. Pro usnadnění lze spravovat tyto aplikace

naráz v projektu pomocí tool-manifestu souboru, který uchovává informace o aplikacích a jejich verzích přímo v projektu. Před nainstalováním aplikací je třeba tento soubor příkazem:

```
dotnet new tool-manifest
```

Následně je pak třeba provést instalaci aplikace pro správu Entity Frameworku příkazem v adresáři s tool-manifestem

```
dotnet tool install --local dotnet-ef --version 6.0.0
```

3.2.1 Dotazování dat v Entity Framework

Entity Framework pro přístup k datům používá objekty implementující rozhraní IQuariable, díky kterému lze prvně objekty vyfiltrovat pomocí LINQ a následně se na data dotázat, čímž nevzniká příliš velký dotaz a následná filtrace dat v kódu. Pro spojování modelů relací “m“ to “n“ je nutno je správně přiřadit. To lze provést pomocí LINQ s rozšiřujícími metody Include() a ThenInclude(). [19]

4 Prvky vlastního systému Firios

Řešení implementovaného systému Firios je postavené na hlavně na webovém serveru a neomezeném množství oznamovačů, takže je otevřen do budoucna pro napojení na další systémy. Pro napojení je třeba znát jeho architekturu.

4.1 Architektura

Aplikace Firios se skládá z pěti hlavních komponentů, zobrazených na obrázku 4.1.

- Webový server - Stará se o správu uživatelských účtů, výjezdů, komunikaci s oznamovači a notifikaci uživatelů.
- Databáze - Datové úložiště pro webový server.
- Oznamovač - Zajišťuje oznámení o výjezdu pomocí REST api webovému serveru.
- Uživatel - Aplikace otevřená na straně uživatele v jeho zařízení.
- Monitor - Speciální typ uživatele pro zobrazování informací na hasičské zbrojnici, či jiném místě, kde je potřeba vidět aktuální stav jednotky.

Webový server slouží jako hlavní uživatelské rozhraní a celá aplikace je postavená na principu webových stránek. Webový server zajišťuje základní správu aplikace stejně jako komunikační rozhraní pro komunikaci s oznamovači a uživateli. Slouží tedy jako prostředník.Pro uživatelské rozhraní server využívá HTML šablony, do kterých vkládá kód podle stránky kterou aktuálně uživatel žádá a pro komunikaci s oznamovači využívá webového API.

Databáze představuje databázový SQL server.

Oznamovač je jakékoliv zařízení, které komunikuje po protokolu HTTPS a posílá data o nových výjezdech dle předem dané struktury ve formátu JSON.

Uživatel je jakékoliv zařízení s podporou prohlížeče, přičemž podpora funkcí se liší dle typu a verze prohlížeče. Slouží pro interakci uživatele s aplikací.

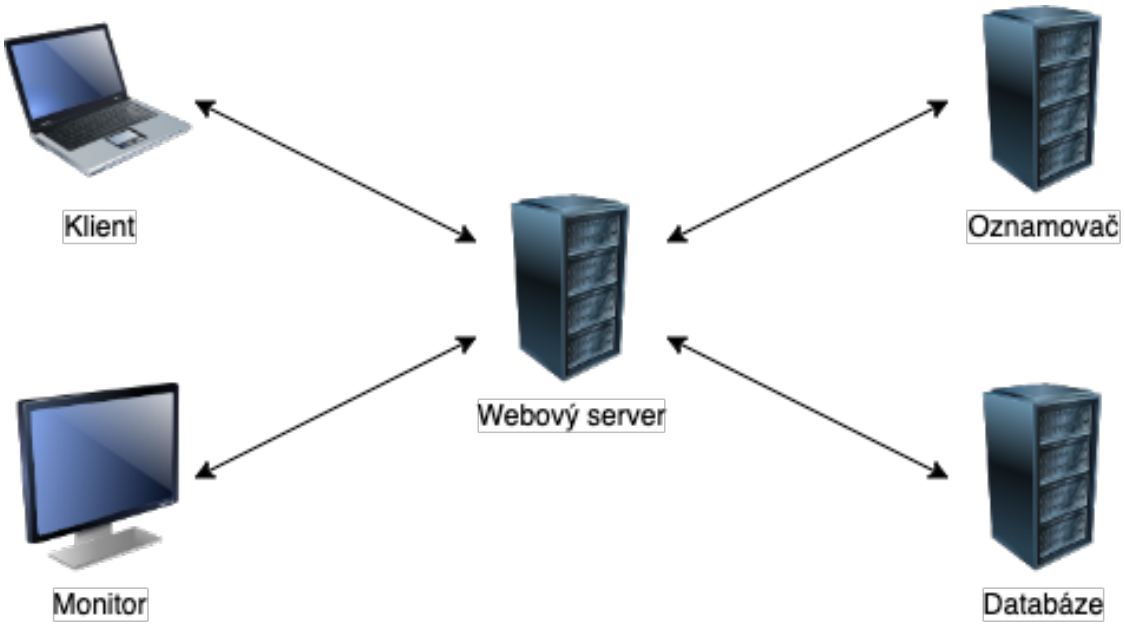
Monitor je speciální typ uživatele s omezenými právy, který slouží pouze prohlížení a nemůže se účastnit výjezdů.

4.2 Webový server

Webový server obstarává tři funkce.

- Uživatelské rozhraní
- Komunikaci s databází
- Webové API

Uživatelské rozhraní je postavené na architektuře MVC, kde každá se podle adresy v HTTP požadavku provolá daný controller, který dle potřeby dotáže databázi



Obr. 4.1: Architektura Firios

o další data, popřípadě rovnou zavolá view, do kterého popřípadě pošle data, která následně view vygeneruje v HTML šabloně.

4.2.1 Controllery

Controllery lze jednoduše vygenerovat pomocí šablon samotného Visual Studia. Stačí zadat třídu uloženou v databázi, podle které se má controller vygenerovat (například UserEntity) spolu s databázovým kontextem sloužícím pro komunikaci s databází a Visual studio samo vygeneruje základní CRUD operace pro danou třídu. Tento controller je možné dále upravovat a přidávat metody pro specifické operace jako je přihlášení, odhlášení, či komplexnější operace.

- HomeController
- UserController
- IncidentController
- IncidentEntitiesController

UserController

Tento controller se stará o základní správu uživatelů. Obsahuje metody pro přihlášení, odhlášení, výpis všech uživatelů, detailní výpis uživatele,vytvoření uživatele, editaci uživatele, mazání uživatele a změnu hesla uživatele.

Při přihlášení se uživateli na základě jeho identifikátoru, aktuálního času a pseudonáhodného čísla vygeneruje identifikátor relace ve formě cookie, která se posílá

s každým dotazem, a tak je možné na adresách, které jsou určeny jen pro určité role, ověření na základě této relace, která je spojena s tabulkou uživatele a jeho rolí. Při odhlášení uživatele se cookie session smaže.

Při výpisu všech uživatelů se prvně provede dotaz na databázi pro tyto data, které se následně pošlou do view šablony a v té se pak pomocí iterací cyklu foreach vypíšou data jednotlivých uživatelů jako řádky tabulky.

Pro detail aplikace vybere jen jednoho uživatele, u kterého následně spojí pomocí spojovací tabulky UserIncident s výjezdy, kterých se uživatel účastnil a stejným principem jako v případě výpisu všech uživatelů se vypíší tyto výjezdy. Metoda pro vytvoření, editování i mazání uživatele je přístupná pouze pro roli Velitel jednotky, která slouží jako administrátor aplikace.

Aplikace definuje pět rolí.

- Velitel jednotky
- Velitel
- Strojník
- Hasič
- Monitor

Role velitele zastupuje hasiče s velitelským kurzem, role strojník hasiče se strojnickým kurzem a role hasič hasiče se základním výcvikem. Tyto role nemají speciální oprávnění a slouží pouze pro informaci hasičů při výjezdu, jelikož je třeba aspoň jeden strojník a velitel.

Role monitor slouží pouze pro přihlášení zobrazovacího zařízení na hasičské zbrojnici a nemůže se účastnit výjezdů.

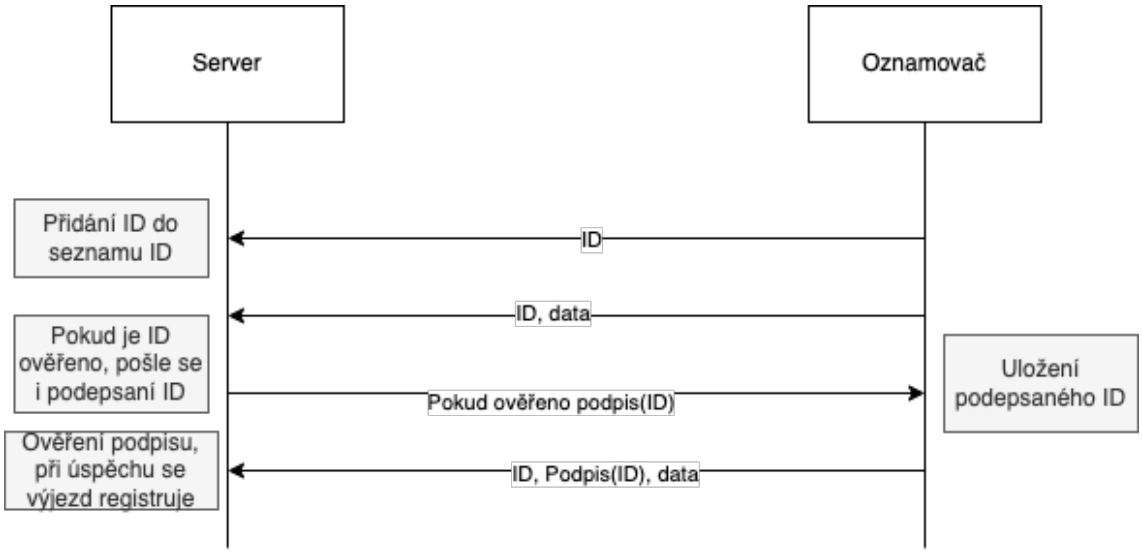
IncidentController

Controller incidentu obsahuje metody pro vypsání všech výjezdů, detailu výjezdu a změnu stavu výjezdu na neaktivní, nebo aktivní, která je umožněna pouze velitelů jednotky.

HomeController

Domovský controller obsahuje metodu index, sloužící pro zobrazení domovské stránky a přesměrování při dotazu na neexitující stránku a vypsání této informace. Další metoda slouží pro vypsání oznamovačů, které po spuštění webového serveru zkoušeli notifikovat o výjezdu, nebo se zaregistrovat.

Při validaci oznamovače velitelem jednotky se vygeneruje podpis uložený v paměti RAM na základě privátního klíče vygenerovaného do konfiguračního souboru, který se pošle oznamovači a následně se validuje pomocí veřejného klíče. Při registraci výjezdu pak není třeba dotazovat data z databáze. Protokol je zobrazen na



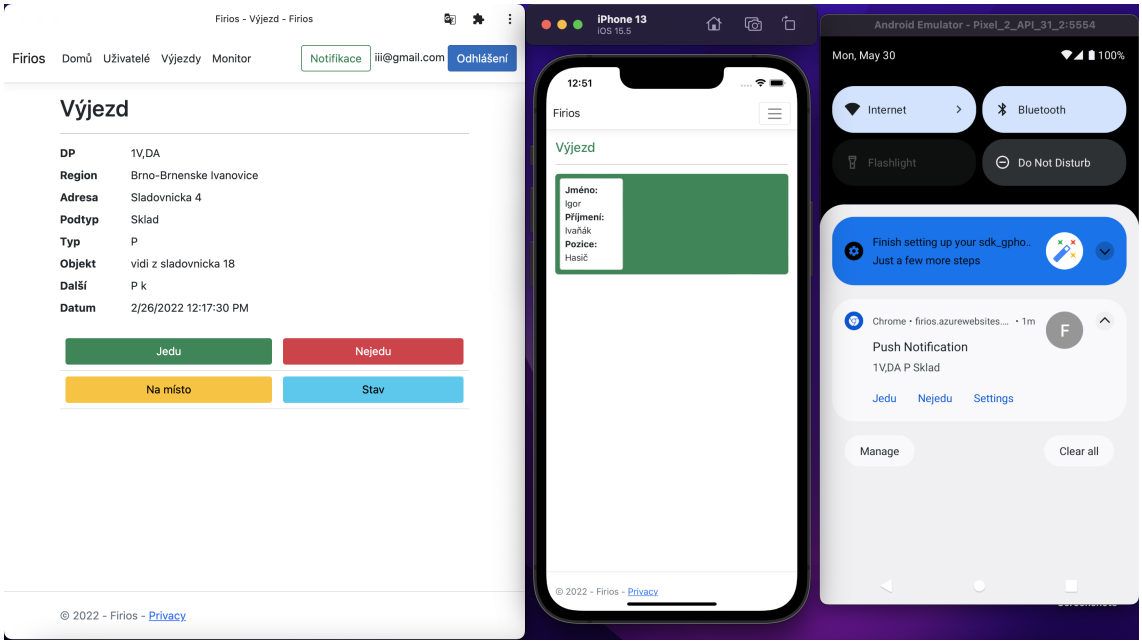
Obr. 4.2: Kryptografické schéma registrace oznamovače

obrázku 4.2. Celý protokol běží pod SSL/TLS protokolem, a tak není třeba data šifrovat ani autentizovat server, protože je již autentizován na základě TLS certifikátu.

Další metoda UserConfirmAction slouží pro přihlášení uživatele k výjezdu. Na tuto stránku vede odkaz z push notifikace a nabízí tři stavy.

- "Jedu"- Znamená, že uživatel se dostaví na hasičskou zbrojnici.
- "Nejedu"- Znamená, že uživatel se výjezdu nemůže účastnit.
- "Na místo"- Znamená, že uživatel není schopen se dostavit na hasičskou zbrojnici, ale dostaví se přímo na místo zásahu.
- "Stav"- Slouží pro přesměrování na stránku monitoru, kde je zobrazeno, kdo jede, nejede, nebo dorazí na místo.

Poslední metodou home controlleru je InteractiveIncident, která slouží pro zobrazení stránky, která ukazuje aktuální stav posledního přijatého výjezdu. Pomocí webových socketů dostává klient informaci od serveru o nových výjezdech, nebo změnách stavu uživatelů u posledního výjezdu. Stav lze zjišťovat pomocí nadpisu stránky přičemž černá znamená ještě nepřipojeno, nebo odpojeno, zelená již připojeno. Na stránce jsou tři kontejnery kam se podle stavu vkládají uživatelé, popřípadě při změně stavu odebírají. Všechny metody controlleru jsou zobrazeny na obrázku 4.3, kontejnery zvlášť na obrázku 4.4.



Obr. 4.3: Výjezd ve Firios (Prohlížeč, iOS, Android)

Obr. 4.4: Zobrazení výjezdu na monitoru

IncidentEntitiesController

Controller IncidentEntitiesController se stará se webové api komunikující s oznamovači a klientským programem v JavaScriptu. Metoda IncidentRegistration slouží pro registraci nového incidentu oznamovačem. Metoda na začátku zkoumá správný formát identifikátoru oznamovače a následně ověřuje, zda je oznamovač důvěryhodný. Pokud je oznamovač důvěryhodný, ale neposlal podepsané ID, server pošle nazpět i toto podepsané ID. Formát zprávy pro oznámení o výjezdu je zobrazen ve výpisu 4.1.

Dále obsahuje controller metodu pro přihlášení uživatele k výjezdu, která se provolává v JavaScriptu při stlačení tlačítka na stránce UserConfirmAction. Registraci webového socketu, který se přidá do singletonu WebSocketFiriosManagerService, který slouží jako úložiště v paměti RAM aplikace. Metoda PushNotificationRegistration od přihlášeného uživatele uloží do databáze data pro následné webové push notifikace. Poslední metoda NotifierRegistration slouží pro registraci oznamovače. Pokud oznamovač pošle pouze své ID a webový server nemá v RAM paměti uložený vygenerovaný podpis, uloží se tento identifikátor do RAM aplikace, pokud webový server již uložil a uživatel ověřil ID, pak server pošle zpět podepsané ID. Pokud oznamovač pošle jak svůj identifikátor, tak podepsaný identifikátor, server ho ověří dle svého klíče a odešle jestli ověření sedí. ID je hexadecimální textový řetězec vygenerovaný náhodně, v základu o délce 128 bitů a podpis je hexadecimální textový řetězec. Formát zprávy pro registraci k výjezdu je zobrazen ve výpisu 4.2

```
{
  "validationId": "96db738a9c09d00e",
  "signatureId": "39d09b09c90a0394",
  "mpd": "1V, DA",
  "region": "Brno-Lisen",
  "address": "Marianske udoli 456",
  "subType": "Strom",
  "type": "P",
  "objectName": "Hriste",
  "additionalInformation": "Oznamovatel hasi vodou",
  "level": "II.st.",
  "isActive": true,
  "date": "2021-09-14T9:45:15.045Z"
}
```

Výpis 4.1: Vstup nového incidentu

```
{
  "validationId": "96db738a9c09d00e",
  "signatureId": "39d09b09c90a0394"
}
```

Výpis 4.2: Vstup registrace oznamovače

4.2.2 Autentizace

Autentizaci v aplikaci řeší hned několik tříd. Pro přihlášení a ohlášení je to třída UserController vysvětlena výše. Pro vizuální stránce se o autentizaci starají třída AuthMiddleware a šablona _Layout.cshtml, do které se vkládají ostatní stránky. AuthMiddleware kontroluje každý požadavek na server a v případě, kdy je v požadavku přítomna cookie s názvem session, tak přidá do pole Items hodnoty dle daného ověření. Liší se dle ověření uživatele a administrátora, který má navíc jednu hodnotu. Až požadovaná stránka dorazí do view, aplikuje se logika v šabloně na vygenerování polí, které jsou potřeba na základě ověření. Pro autentizaci určitých částí aplikace je na každé metodě, která autentizaci potřebuje provolána funkce z objektu FiriosUserAuthenticationService, která ověří zda má uživatel potřebná práva. Ověřování uživatelů v celé aplikaci je postavena na ověřování cookie session v databázi a navázání této tabulky na tabulku uživatelů, která obsahuje pozice. Hesla jsou uložena ve formě hashe za použití algoritmu SHA256. Použita je kryptografická sůl i iterace.

Třída FiriosUserAuthenticationService obsahuje metody, které vracejí uživatele na základě session, nebo objektu request a metody, které uživatele validují na základě session, request, nebo objektu uživatele a kolekce stringů definující pozice.

Třída FiriosSourceAuthenticationService prvně načítá soukromý klíč z konfiguračního souboru a kromě pomocných metod na transformaci textového řetězce do pole bajtů a obráceného procesu definuje metody pro generaci podpisu, vybrání podpisu z kolekce a validaci podpisu.

4.2.3 Services

Kromě výše zmíněných služeb aplikace obsahuje IncidentIdService třídu zaregistrovanou jako singleton, která slouží pouze pro uložení posledního výjezdu, UserHelperService třídu, která obsahuje komplexnější logiku pro uložení uživatele k výjezdu a selekci výjezdu na základě ID. Poslední službou je WebSocketManagerService, která ukládá kolekci webových socketů.

4.3 Klientská část

Aplikace by nebyla schopna pracovat bez použití klientského programovacího jazyka JavaScript. Tento jazyk umožňuje prohlížeči interagovat se serverem a upravovat HTML dokument bez potřeby nového dotazu webového serveru.

4.3.1 Základní logika aplikace

Na každé stránce se načítá automaticky soubor site.js. Ten obsahuje metodu pro získání hodnoty cookie a metodu na získání relace reprezentované cookie se jménem "Session". Potom jsou v souboru pomocné metody pro práci s tlačítkem označujícím stav webových push notifikací. Tlačítko má 4 stavy:

- Modré - Probíhá práce v pozadí.
- Zelené - Znamená že uživatel se výjezdu nemůže účastnit.Vše proběhlo úspěšně a notifikace fungují.
- Žluté - Notifikace nejsou povoleny, ale prohlížeč je podporuje.
- Červené - Notifikace jsou buď zakázané, nebo nejsou prohlížečem podporovány, nebo došlo k chybě při spojení s webovým serverem.

Tyto stavy se mění na základě přítomnosti "serviceWorker"v objektu navigator, poté se vyčká na načtení a zkusí se zaregistrovat service worker a na základě úspěšnosti vyžádání povolení k notifikacím se tlačítko zbarví. Dle úspěšnosti se také provede odeslání údajů k webovým push notifikacím, popřípadě se zažádá o povolení uživatelem. Na konci souboru je skript který si ukládá kopii cookie do local storage, pro případ kdy si prohlížeč cookie uzná jako vypršenou. To se děje, protože cookie nemá nastavený expiration. V rámci aplikace však tato hodnota nemá smysl, protože v případě odhlášení uživatele z hlediska času, kdy se nepřihlásil by znamenalo potíže u člena, který by aplikaci pravidelně neotevíral.

4.3.2 Service worker

Service worker funguje jako proxy pro prohlížeč. Dokáže zachytávat requesty na základě event listenerů. To lze využít pro ukládání stránek, aby aplikace fungovala i bez připojení k internetu. Pro implementaci webových push notifikací potřebuje service worker metodu, která se zavolá při eventu push a metodu, která se zavolá při eventu notificationonclick. Pro push event se příchozí data vypíší do textu notifikace. Pro event notificationonclick se uživatel přesměrovává na stránku aplikace, nebo se odesílá rovnou registrace.

4.3.3 Monitor

Pro zobrazení aktuálního stavu členů jednotky musí webový server notifikovat klienta. Pro tuto funkcionalitu byly využity webové sockety. Webové sockety eliminují nevýhodu bezstavového protokolu HTTP a přidávají mu možnost stav mít. Při requestu si server počká s odpovědí a odpoví až bude potřebovat.

Webovým socketem může aplikaci dojít status v případě, kdy se klientská strana registrovala, nový výjezd, nebo změna u uživatele. Stránka následně reaguje na změnu vymazáním aktuálních dat, nebo změnou elementu konkrétního uživatele. Pro případ, kdy je spojení přerušeno, se webový socket snaží znovu připojit.

4.4 Implementace Entity Frameworku

Pro aplikaci byly zvoleny tři základní modely. Jedná se o model incidentu, neboli události, model uživatele a model pro uchování čísla relace a dat k webovým push notifikacím.

Uživatel obsahuje základní informace o členu jednotky, jako je jméno, příjmení, tituly, pozice v jednotce, přihlašovací heslo ve formě hashe, kryptografické soli a počtu iterací, nebo email.

Model incidentu obsahuje naopak základní data u události, jako je typ události, jméno objektu, kterého se událost týká, doplňkové informace, datum, stupeň, její aktuální stav a další informace. Zároveň musí model uživatel udržovat, jakých incidentů se účastnil a naopak model události udržovat informace o tom, kteří uživatelé se výjezdu zúčastnili, nebo nezúčastnili. Tato vazba je právě vazbou “n“ na “m“ a je potřeba ji upravit tak, aby ji Entity Framework pochopil a správně uložil do databáze. Byla navržena speciální třída UserIncidentEntity, obsahující identifikační číslo uživatele, incidentu, odkazy na jejich uložení v paměti a jejich stav k události ve formě textového řetězce. Zároveň je nutno Entity Framework o vazbě informovat. To lze buď ve třídě dědící ze třídy DbContext přepsáním metody OnModelCreating, nebo atributy přímo v samotném modelu entity prostřednictvím atributu ForeignKey. Vazba uživatele a modelu pro relaci spolu s daty pro webové push notifikace je pouze 1 na n a tak je při této vazbě daleko menší komplexita. Schéma vazem modelů je zobrazeno na obrázku 4.5. Třída dědící ze třídy DbContext je potřebná pro správu databáze. V souboru Program.cs je třeba ji zaregistrovat a vložit ji textový řetězec pro připojení k databázi.

Po vytvoření výše uvedených tříd, lze vytvořit migrace pomocí dotnet-ef. Migrace slouží pro držení stavu aktuálních dat oproti databázi. Při aktualizování databáze pak Entity Framework ví, se kterou verzí dat databáze pracuje, a zda je potřeba přidat pole, nebo odebrat. V obou případech se musí řešit, jakým způsobem se

Obr. 4.5: Schéma entit

budou pole přidávat, nebo odebírat. Při přidávání je třeba definovat, zda se dříve vyplněná data mají nastavit na konkrétní hodnotu, nebo nabývat prázdné hodnoty.

Pro vytvoření migrace lze použít následující příkaz:

```
dotnet ef migrations add InitialMigration
```

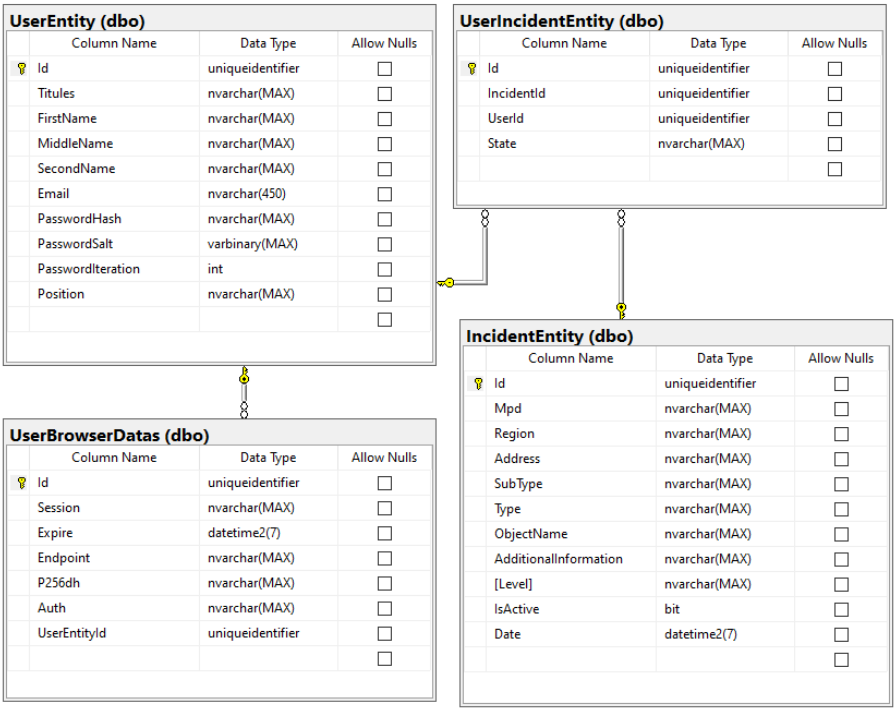
Po vytvoření migrací lze vytvořit databázi příkazem:

```
dotnet ef database update
```

Entity Frameworku po provedení update zajistí vytvoření databáze. Schéma databáze je zobrazeno na obrázku 4.6. Pro její prohlížení lze využít například aplikace SQL Server Management Studio, která umožňuje správu databáze, výpisy a úpravu dat, předvytvoření selekce z tabulek a další.

4.5 Pomocný konzolový nástroj

Webový server vyžaduje kryptografickou konfiguraci. Musí se vygenerovat VAPID pár pro web push API a soukromý klíč pro podepisování identifikátorů oznamovačů. Pro ulehčení práce nasazovatele byla vytvořena konzolová aplikace pro generaci těchto údajů. Stačí zapnout EXE soubor a do konzole se vypíše JSON objekt, který nasazovatel nakopíruje do konfiguračního souboru appsettings.json.



Obr. 4.6: Schéma databáze

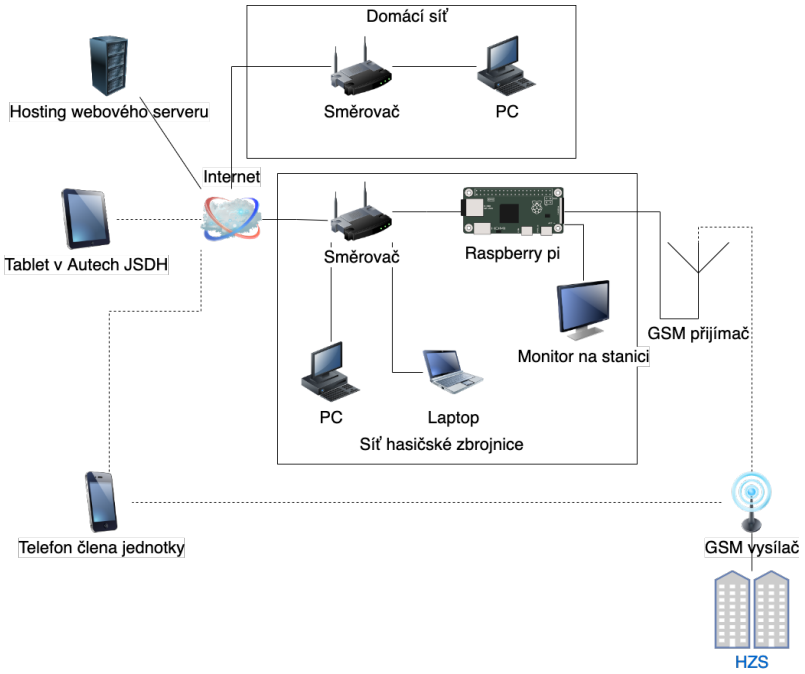
4.6 SMS oznamovač

Jako oznamovač byl vyvinut kód v programovacím jazyku python. Jedná se o jednoduchý skript řešící přeposílání SMS zpráv na webový server. Obsahuje metody na přeformátování SMS zprávy na JSON objekt, ukládání a načítání konfigurace z konfiguračního souboru, vytvoření pseudonáhodného řetězce, registraci u webového serveru, zasílání AT příkazu po sériové lince GSM modulu a posílání HTTPS dotazů na webová server. Na začátku programu se do sériové linky zapíše trojice příkazů, která slouží pro inicializaci, nastavení GSM modulu do textového režimu a módu, kdy GSM modul přijatou SMS zprávu rovnou posílá po sériové lince. Poté program čeká na příchozí zprávy ze sériové linky a pokud zpráva obsahuje kód přijaté zprávy a obsahuje číslo z konfigurace, pak se zpráva naformátuje na objekt JSON a pošle na webový server spolu s identifikátorem.

Program by měl být spuštěn jako služba v operačním systému, aby se například nemusel zapínat při každém zapnutí operačního systému, což může mít velký dopad v případě výpadku proudu. V rámci bakalářské práce byl vytvořen soubor firios.service, který obsahuje konfiguraci služby. Soubor je třeba uložit do adresáře /etc/systemd/system a po obnovení konfigurace pomocí "systemctl daemon-reload"lze se službou pracovat pomocí systemctl příkazů.

5 Nasazení vlastního řešení

Pro zprovoznění celé aplikace je třeba nasadit správně všechny její části. Je důležité změnit základní nastavení kvůli bezpečnosti kryptografických údajů a nakonfigurovat správnou URL adresu pro oznamovače. Celá infrastruktura je zobrazena na obrázku 5.1



Obr. 5.1: Síťová architektura aplikace Firios

5.1 Nastavení kryptografických klíčů

Před samotným publikováním aplikace je vhodné a důležité přepsat aplikaci kryptografické klíče v konfiguračním souboru appsettings.json, aby byly klíče unikátní. Pro zjednodušení byla vytvořena konzolová aplikace, kterou stačí pouze spustit a na výstupu vypíše nově vygenerované klíče i ve formátu JSON. Stačí tedy pouze přepsat sekci obsahující klíče v souboru appsettings.json výstupem konzolové aplikace.

5.2 Publikace webové aplikace

Aplikace je postavená na multiplatformním frameworku, takže lze hostovat na celé řadě systémů. Jako hostovací řešení bylo zvoleno hostování u firmy Microsoft na jejich cloudovém řešení Azure.

5.2.1 Nasazení webového serveru

Visual Studio rovnou nabízí možnost publikování aplikace do Azure. Takže při kliknutí na projekt je mezi možnostmi Publish. Otevře se nabídka, kde se zvolí možnost Azure a na další nabídce Azure App Service (Windows). Dále je potřeba vytvořit App Service Instanci, do které aplikaci umístíme. Otevře se další dialogové okno s volbou názvu resource group, lokace služeb a plán, který určuje parametry a cenu hostovaného stroje. Po vytvoření instance se okno zavře a konfiguruje se opět samotná aplikace, kde se musí navolit nově vytvořená instance. V okně konfigurace API managmentu pro tuto aplikaci nebylo třeba nic zadávat, a tak bylo zvoleno skip this step. Na poslední části byla ponechána volba pubxml souboru [20].

5.2.2 Vytvoření nové databázové služby

Po vytvoření služby pro aplikaci Visual Studio zobrazí možnost konfiguraci závislostí. Jedna z těchto závislostí je SQL Server Database, která slouží jako databáze pro aplikaci a konfiguruje se tlačítkem configure. Pro aplikaci byla zvolena možnost Azure SQL Database, vytvořena nová databáze nazvaná firios_db umístěná do stejné resource group. Před vytvoření databáze je třeba vytvořit instanci databáze zvolením new u pole database server. Zde lze zvolit jméno, lokace, administrátorské jméno a heslo. Po konfiguraci instance lze již vytvořit celou databázi. Po vytvoření databáze se musí dokonfigurovat aplikace zvolením nově vytvořené databáze a zadáním přihlašovacích údajů, díky kterým dialogové okno vygeneruje connection string, který se používá pro připojení na straně aplikace. Na posledním kroku okna bylo ponecháno NuGet packages i Secrets store.

5.2.3 Předání migrací databázi

Po vytvoření aplikační služby i databáze je třeba předat databázi migrace, podle kterých se vytvoří tabulky. Na stránce publish zvolením tlačítka edit lze v nastavení Entity Framework migrations vybrat aplikování zvolených migrací. Pak už stačí jen kliknout na tlačítko publish a povolit v portálu Azure u aplikace webové sockety v settings -> configuration -> general settings.

5.3 Nastavení webové aplikace

Po nasazení webové aplikace je třeba provést základní nastavení. Na stránce je třeba kliknout na tlačítko login. V případě kdy ještě není zaregistrovaný žádný uživatel, přesměruje se tato stránka na registraci nového uživatele, kterého výjimečně povolí

a automaticky nastaví prvního uživatele na velitele jednotky, který slouží jako administrátor. Dále je potřeba jen přidat další uživatele a nasadit oznamovač, popřípadě účet monitoru.

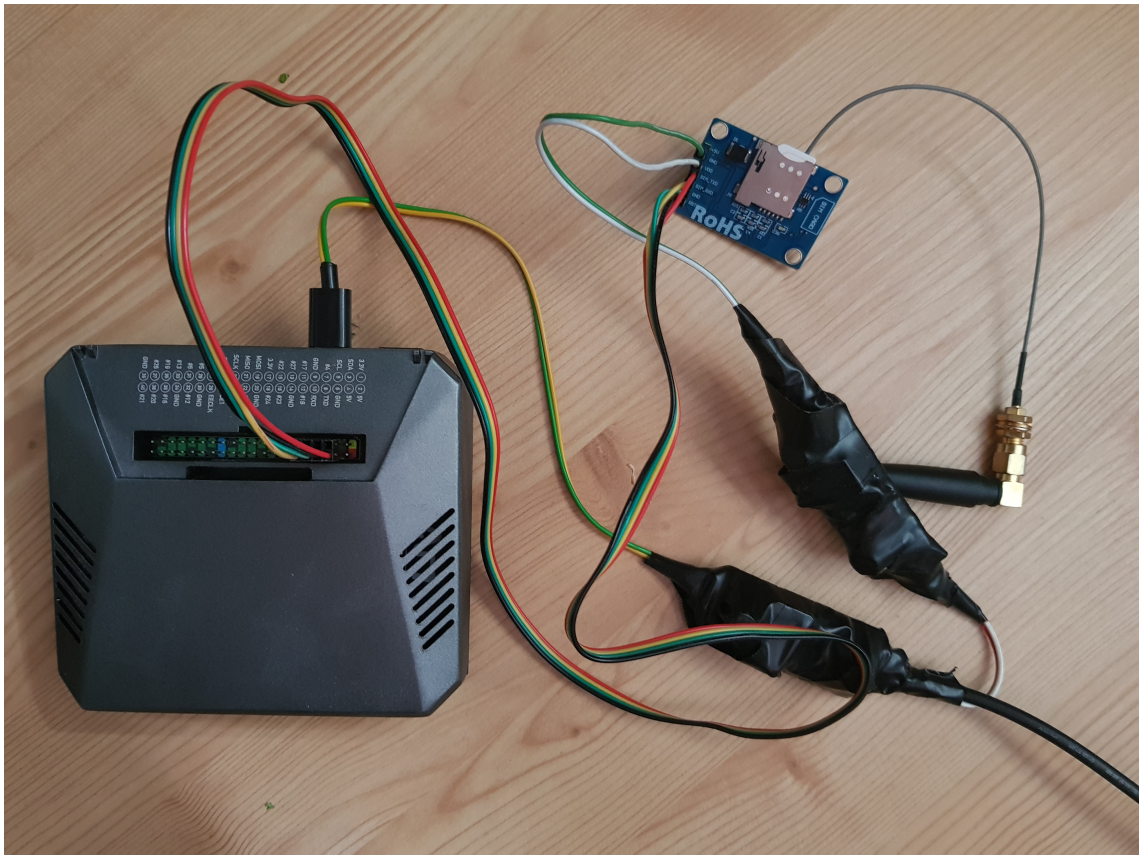
5.4 Nasazení oznamovače

Nasazení oznamovačů se liší dle typu oznamovače. V rámci této bakalářské práce byl vytvořen pouze jeden, který čte SMS zprávy z čísel a ty následně převede do formátu JSON a posílá na webový server.

Lze vytvořit i další, a tak je aplikace otevřená do budoucna pro notifikování hasičským záchranným sborem rovnou pomocí internetové sítě, což by celý proces výrazně urychlilo oproti SMS zprávám. Avšak pro zachování možnosti notifikování i bez sítě internet by mělo být zachováno svolávání pomocí SMS.

5.4.1 Nasazení SMS oznamovače

Pro nasazení SMS oznamovače je třeba mít předem připravené Raspberry Pi s nainstalovaným systémem Raspbian, nakonfigurovanou konektivitou k síti internet a povolený sériový port se zakázaným debug módem [21]. Dále je potřeba samotný GSM modul. Poté je třeba vytvořit nového uživatele firios, přidat ho do skupiny dialout, následně přesunout soubor firios.service do adresáře /etc/systemd/system/ [22] [23] a soubor firios.py do adresáře /etc/firios a nastavit vlastnictví na uživatele firios a skupinu firios. Následně znovu načíst služby příkazem "sudo systemctl daemon-reload", povolit automatické spouštění služby po startu systému pomocí "sudo systemctl enable firios.service"a zapnout službu pomocí "sudo systemctl start firios.service". Aplikace si sama vygeneruje soubor configuration.json v adresáři /etc/firios/, který lze editovat pro povolení SMS z dalších čísel, změnu webového serveru, nebo změnu ID, je ale potřeba službu restartovat. Nakonfigurované Raspberry Pi je zobrazeno na obrázku 5.2.



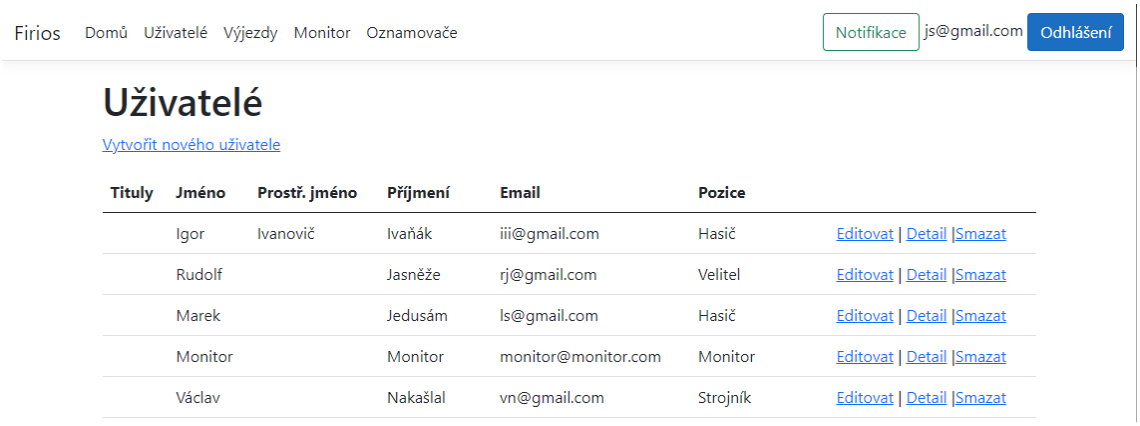
Obr. 5.2: Raspberry Pi jako oznamovač a monitor

6 Uživatelské rozhraní

Po nasazení všech částí aplikace mohou členové aplikaci otevřít v jakémkoliv prohlížeči, nebo dokonce nainstalovat jako progresivní webovou aplikaci přidáním na domovskou obrazovku.

6.1 Přihlášení uživatele

Když uživatel navštíví stránku, jako první se potřebuje přihlásit na přihlašovací stránce údaji obdrženými od správce systému, které daný účet vytváří. Heslo lze samozřejmě později změnit a to po rozkliknutí své emailové adresy v horní liště, která je vidět na obrázku 6.1, zadáním správného hesla a nového hesla splňujícím požadavky na komplexitu. Každý uživatel si může změnit pouze své heslo, kromě administrátora, který je schopen upravovat všechny informace.

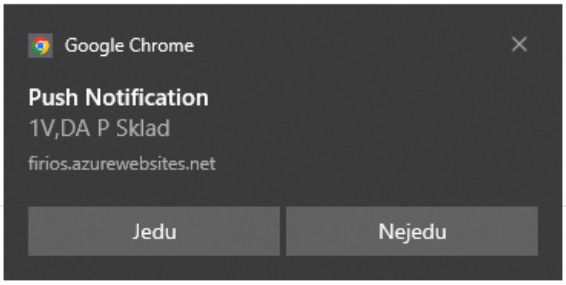


Obr. 6.1: Výpis uživatelů

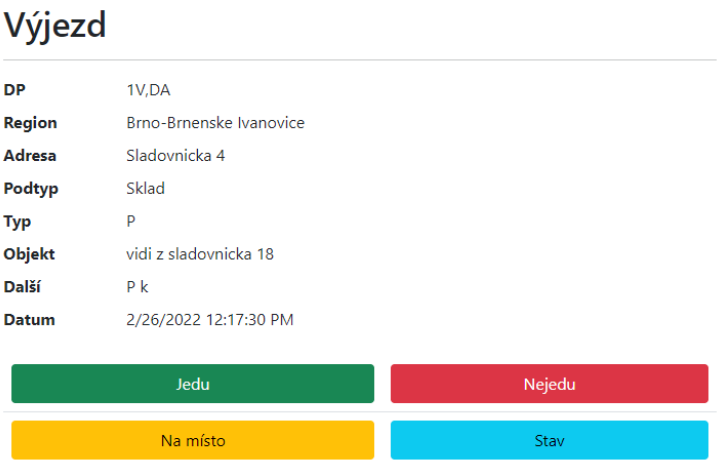
6.2 Oznamovače, notifikace a monitor

Uživatel má k dispozici zapnutí notifikací na výjezdy 6.2a. Notifikace obsahuje dvě tlačítka, která rovnou odešlou stav uživatele na webový server a po kliknutí na samotnou notifikaci se zobrazí stránka s akcemi pro uživatele 6.2b.

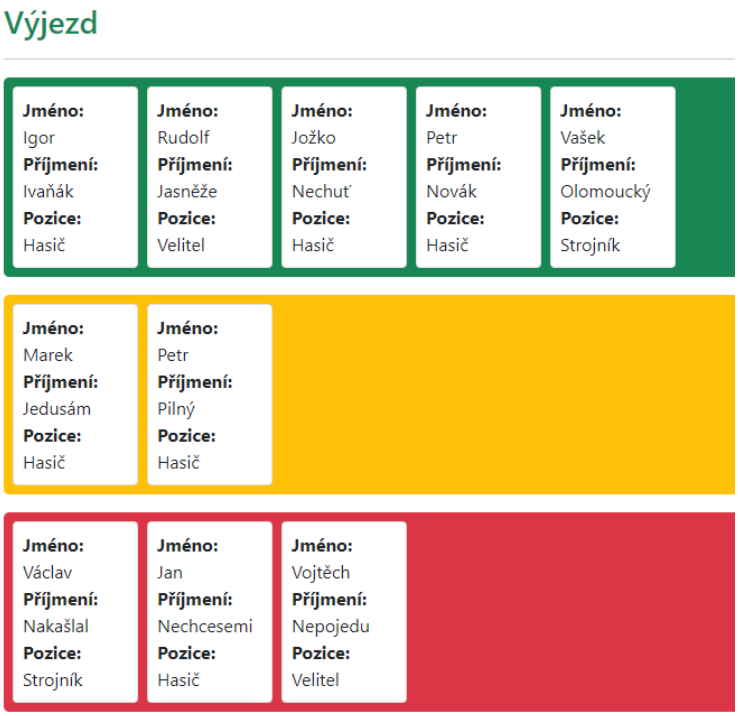
Tlačítko "Stav"odkazuje stejně jako tlačítko "Monitor"v horním menu aplikace odkaz na stránku určenou převážné pro účet monitor 6.2c. Tato stránka slouží pro zobrazení aktuálního stavu a její hlavní účel je neustálé spuštění na hasičské stanici, aby před odjezdem byli přítomni všichni hasiči. Tyto notifikace vyvolává webový



(a) Notifikace pro prohlížeč chrome



(b) Potvrzovací stránka



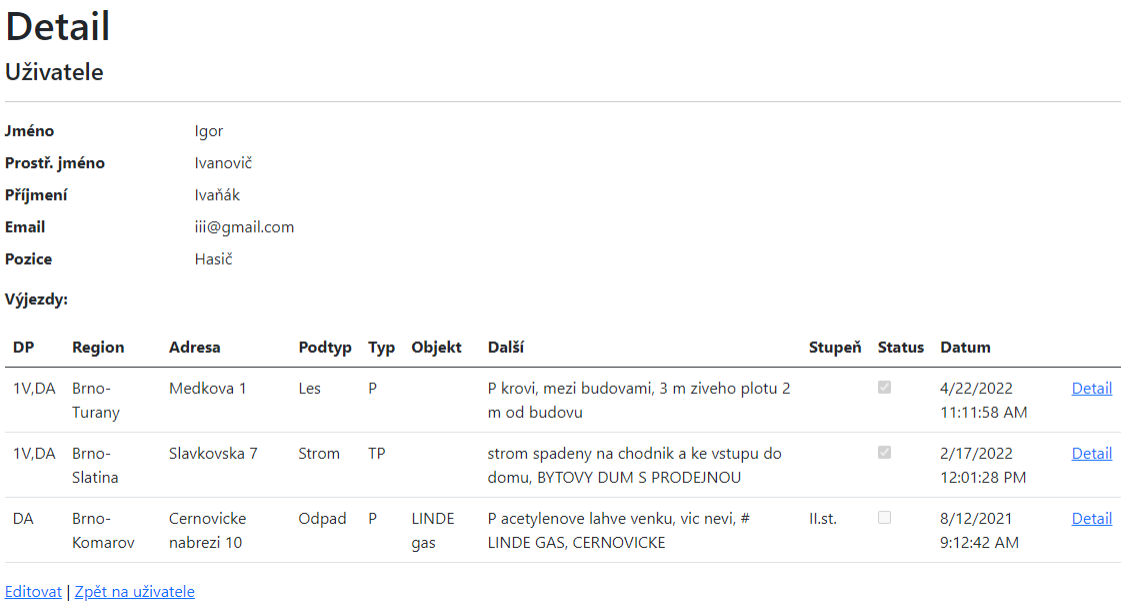
(c) Stránka monitoru

Obr. 6.2: Potvrzovací rozhraní s monitorem

server při oznámení od oznamovače. Oznamovač posílá nové výjezdy webovému serveru. Oznamovače má na starost pouze administrátor a musí oznamovač povolit na stránce oznamovače.

6.3 Sekce uživatelů

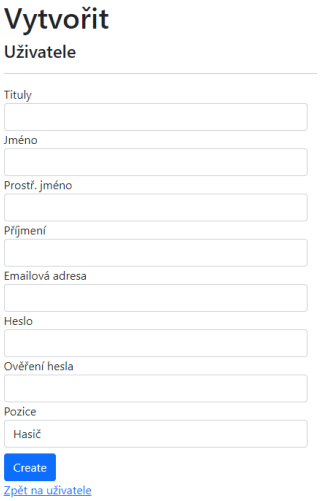
Po kliknutí na tlačítka "Uživatelé", lze zobrazit všechny uživatele v aplikaci6.1. Stránka pro detaily 6.3 je přístupná pro všechny uživatele a obsahuje kromě základních údajů o uživateli i výjezdy na kterých se účastnil. Pod nadpisem stránky je tlačítko směrující na vytvoření nového uživatele 6.4. Toto tlačítko je přístupné stejně jako tlačítko "Editovat"a "Smazat"u každého uživatele pouze pro administrátora. Stránka na editaci se podobá principem vytvoření uživatele a stránka na mazání obsahuje pouze potvrzovací tlačítko.



Obr. 6.3: Detail uživatele

6.4 Sekce výjezdů

Sekce výjezdů obsahuje tak jako uživatelská sekce přehled všech výjezdů 6.5. Po rozkliknutí na detail, který je taktéž jako v uživatelské sekci přístupný všem uživatelům, lze vidět detailní informace o výjezdu spolu s účastněnými hasiči 6.6. Mezi detailem výjezdu a hasiči se administrátorovi zobrazuje tlačítko na uzavření/otevření registrace k výjezdu.

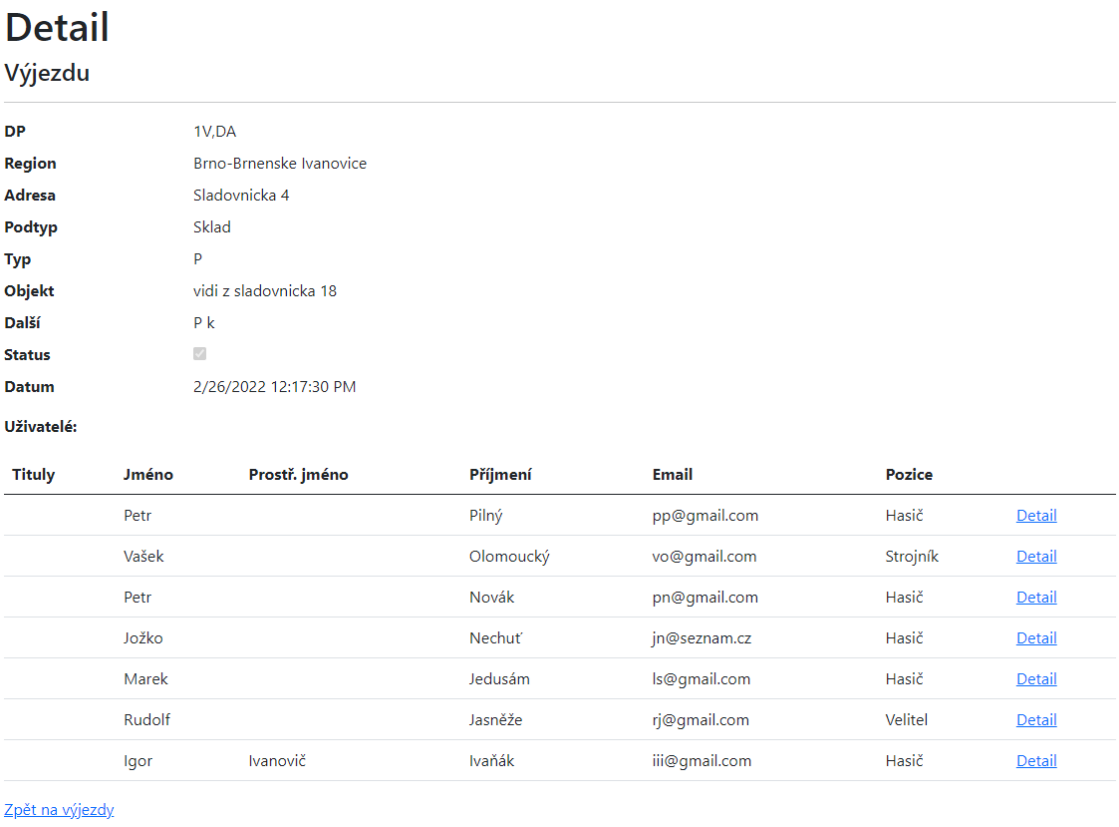


Obr. 6.4: Vytvoření nového uživatele

Výjezdy

DP	Region	Adresa	Podtyp	Typ	Objekt	Další	Stupeň	Status	Datum	
1V,DA	Brno-Turany	Medkova 1	Les	P		P krovi, mezi budovami, 3 m ziveho plotu 2 m od budovu		☑	4/22/2022 11:11:58 AM	Detail
1V,DA	Brno-Slatina	Slavkovska 7	Strom	TP		strom spadeny na chodnik a ke vstupu do domu, BYTOVY DUM S PRODEJNOU		☑	2/17/2022 12:01:28 PM	Detail
DA	Brno-Komarov	Cernovicke nabrezi 10	Odpad	P	LINDE gas	P acetylenove lahve venku, vic nevi, # LINDE GAS, CERNOVICKE	II.st.	☐	8/12/2021 9:12:42 AM	Detail
1V,DA	Brno-Lisen	Marianske udoli 1a	Cerpani	TP	restaurace eldorado	rozvodneny potok, hladina v potoce se zveda a ohrozuje		☐	6/23/2021 3:03:14 PM	Detail

Obr. 6.5: Výpis výjezdů



Obr. 6.6: Detail výjezdu

Závěr

Byla vyvinuta webová aplikace pro dobrovolné hasiče plnící úlohu oznamování nových výjezdů členům jednotky. Aplikace je nasazena na cloudové službě Microsoft Azure. Aplikace obsahuje základní správu uživatelských účtů.

Pro konfiguraci byla vytvořena konzolová aplikace generující kryptografické klíče pro individuální nasazení.

Aplikace obsahuje vstupní webové aplikační rozhraní pro nové možnosti oznámení o výjezdech. Pro oznámení nového výjezdu z SMS byla vytvořena služba v operačním systému Raspbian. Tato služba čte příchozí SMS zprávy z GSM modulu a formátuje je na JSON data, která následně posílá webovému serveru.

Aplikace je zabezpečena ověřením oznamovacího zařízení i uživatelských hesel a identifikátorů jejich relací. Uživatelská hesla jsou uložena ve formě hashe. Aplikace pro hashování využívá kryptografickou sůl i počet iterací.

Aplikace je připravena pro nasazení na hasičské zbrojnici pro uživatele operačních systémů Windows, Linux, macOS i Android. Pro systém iOS nejsou aktuálně podporované notifikace o výjezdech. Operační systém iOS aktuálně nepodporuje webové push notifikace, avšak v lednu 2022 bylo zahájeno jejich testování.

Jako rozšíření aplikace se nabízí napojení Hasičského záchranného sboru České republiky na webové aplikační rozhraní pro rychlejší notifikaci, možnost více jednotkám využívat jednu nasazenou aplikaci, připojení IoT zařízení pro odblokování alarmu, zapnutí kompresoru, zapnutí osvětlení, odemčení dveří.

Literatura

[1] Manel Mena, Antonio Corral, Luis Iribarne, and Javier Criado. A progressive web application based on microservices combining geospatial data and the internet of things. *IEEE Access*, 7:104577–104590, 2019. doi:10.1109/ACCESS.2019.2932196.

[2] Parbat Thakur. Evaluation and implementation of progressive web application. 2018.

[3] Sean Amarasinghe. *Service worker development cookbook*. Packt Publishing Ltd, 2016.

[4] Abhi Gambhir and Gaurav Raj. Analysis of cache in service worker and performance scoring of progressive web application. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, pages 294–299, 2018. doi:10.1109/ICACCE.2018.8441715.

[5] Phakpoom Chinprutthiwong, Raj Vardhan, GuangLiang Yang, Yangyong Zhang, and Guofei Gu. The service worker hiding in your browser: The next web attack target? In *24th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '21, page 312–323, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3471621.3471845.

[6] Hans Fangohr. A comparison of c, matlab, and python as teaching languages in engineering. In *International Conference on Computational Science*, pages 1210–1217. Springer, 2004.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Rfc 2616, hypertext transfer protocol – http/1.1, 1999. URL: http://www.rfc.net/rfc2616.html.

[8] R.M. Adler. Distributed coordination models for client/server computing. *Computer*, 28(4):14–22, 1995. doi:10.1109/2.375173.

[9] Yasuhiko Minamide. Static approximation of dynamically generated web pages. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 432–441, New York, NY, USA, 2005. Association for Computing Machinery. doi:10.1145/1060745.1060809.

[10] Mo Guan and Minghai Gu. Design and implementation of an embedded web server based on arm. In *2010 IEEE International Conference on Software Engineering and Service Sciences*, pages 612–615, 2010. doi:10.1109/ICSESS.2010.5552275.

[11] Rami Vemula. *Real-Time Web Application Development: With ASP. NET Core, SignalR, Docker, and Azure*. Apress, 2017.

[12] Alexey Melnikov and Ian Fette. The WebSocket Protocol. RFC 6455, December 2011. URL: https://rfc-editor.org/rfc/rfc6455.txt, doi:10.17487/RFC6455.

[13] Microsoft. Dokumentace k asp.net, 2021. URL: https://docs.microsoft.com/cs-cz/aspnet/core/?view=aspnetcore-6.0.

[14] Adam Freeman. *Pro Asp. net core MVC*. Apress, 2016.

[15] Andrew Lock. *ASP. NET Core in Action*. Simon and Schuster, 2021.

[16] Steve Smith. Overview of asp .net core mvc. *Microsoft. Last modified January*, 7, 2018.

[17] Atul Adya, José A. Blakeley, Sergey Melnik, and S. Muralidhar. Anatomy of the ado.net entity framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, page 877–888, New York, NY, USA, 2007. Association for Computing Machinery. doi:10.1145/1247480.1247580.

[18] Julia Lerman. *Programming Entity Framework: Building Data Centric Apps with the ADO. NET Entity Framework*. "O'Reilly Media, Inc.", 2010.

[19] Joseph Rattz. *Pro LINQ: Language Integrated Query in C# 2008*. Apress, 2008.

[20] Rahul Sahay. App services. In *Microsoft Azure Architect Technologies Study Companion*, pages 585–621. Springer, 2020.

[21] Configuring the gpio serial port on raspbian jessie and stretch including pi 3 and 4, 2016. URL: https://spellfoundry.com/2016/05/29/configuring-gpio-serial-port-raspbian-jessie-including-pi-3-4/.

[22] Philip Kirkbride. *systemd*, pages 221–234. Apress, Berkeley, CA, 2020. doi:10.1007/978-1-4842-6035-7_11.

[23] David Both. *systemd*, pages 379–410. Apress, Berkeley, CA, 2020. doi:10.1007/978-1-4842-5455-4_13.

Seznam symbolů a zkratek

.NET	Network Enabled Technologies
1V	První Vůz
API	Application Programming Interface
ARM	Typ procesoru založený na RISC architektuře
ASP	Active Server Pages
AUTH	Authentication
CLI	Command-Line Interface
CRUD	Create, Read, Update, Delete
DA	Dopravní Automobil
DB	Databáze
DELETE	Typ metody protokolu HTTP
DH	Kryptografický algoritmus Diffie Hellman
DoS	Denial-of-Service
DP	Dopravní prostředky
EF	Entity Framework
EXE	Windows executable
GET	Typ metody protokolu HTTP
GPIO	General-Purpose Input/Output
GSM	Global System for Mobile communication
HDMI	High Definition Multimedia Interface
HTML	Hypertext Markup Language
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hyper Text Transfer Protocol
ID	Identifikátor

IIS	Internet Information Services
iOS	iPhone OS
IoT	internet of things
IP	Internet Protocol
ISO/OSI	Obecný síťový model organizace ISO
ISO	International organization of Standardization
JSON	JavaScript Object Notation
KOPIS	Krajské operační a informační středisko
LINQ	Language Integrated Query
M2	Typ rozhraní pro připojení disků
MacOS	Operační systém pro počítače Macintosh
MVC	Model View Controller
NB	Nízká Budova
NVMe	Nonvolatile Memory express
ORM	Object-relational mapping
OSI	Open System Interconnection
POST	Typ metody protokolu HTTP
PUT	Typ metody protokolu HTTP
PWA	Progressive Web Application
P	Požár
RAM	Random-access memory
REST	Representational State Transfer
RISC	Reduced Instruction Set Computer
SHA256	Secure Hash Algorithm 256-bit
SIM	Subscriber Identity Module

SMS	Short Message Service
SQL	Structured Query Language
SSD	Solid-state drive
SSL	Secure Sockets Layer
SSMS	SQL Server Management Studio
TCP/IP	Síťový model založený na protokolech TCP a IP
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TP	Technická Pomoc
UDP	User Datagram Protocol
URL	Uniform resource locator
USB	Universal Serial Bus
VAPID	Voluntary Application Server Identity
XML	Extensible Markup Language
YAML	YAML Ain't Markup Language

A Obsah elektronické přílohy

Celá aplikace je v elektronické příloze a adresářová struktura je popsaná níže. Příloha obsahuje i obraz disku Raspberry Pi.

- `/` — kořenový adresář přiloženého archivu
  - `Firios` — Řešení v .NET
    - `Firios/` — Webová aplikace
    - `FiriosSetupApp/` — Pomocný konzolový nástroj
    - `Firios.sln`
  - `FiriosSetupApp/` — Zkompilovaná konzolová aplikace
    - `FiriosSetupApp.exe`
  - `Notifier/` — SMS oznamovač
    - `configuration.json`
    - `firios.py`
    - `firios.service`
  - `RaspberryPi/`
    - `RaspberryPi.txt` — Soubor s odkazem na obraz disku Raspberry Pi