STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Informační systém pro organizátory a návštěvníky kulturního festivalu

Martin Ďuriš
Pardubický kraj

Pardubice 2020

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor č. 18: Informatika

Informační systém pro pořadatele a návštěvníky kulturního festivalu

Information system for organizer and visitors of the cultural festival

Autoři: Martin Ďuriš
Škola: DELTA – Střední škola informatiky a ekonomie, s.r.o.
Ke Kamenci 151, 530 03 Pardubice
Kraj: Pardubický kraj
Konzultant: RNDr. Jan Koupil, Ph.D.

Pardubice 2020

Prohlášení

Prohlašuji, že jsem svou práci SOČ vypracoval/a samostatně a použil/a jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Prohlašuji, že tištěná verze a elektronická verze soutěžní práce SOČ jsou shodné.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Pardubicích dne 12. 3. 2020 ……
Martin Ďuriš

Poděkování

Tímto bych chtěl velmi poděkovat RNDr. Jan Koupilovi, Ph.D. za odborné vedení a pevné nervy při konzultacích práce SOČ.

Anotace

Tato práce SOČ se zabývá tvorbou informačního systému pro organizátory a návštěvníky kulturního festivalu. Dává možnost organizátorům spravovat data pro „návštěvnickou aplikaci“, ve které se návštěvníci mohou dozvědět aktuální informace o průběhu festivalu nebo mají možnost hodnotit jednotlivé interprety.

Práce popisuje vývoj projektu na moderních platformách jako jsou Firebase, Serverless vývoj apod. a dokumentuje způsoby jeho zabezpečení vůči kybernetickým útokům.

Klíčová slova

Backend-as-a-service; GraphQL; Firebase; Serverless; Flutter

Annotation

This project describes the development of information system for organizers and visitors of a cultural festival. Within the system, organizers are able to manage the data for the visitor application. The visitor application provides actual information about the festival for visitors and also gives them the possibility to rate individual performers.

The development using modern platforms and approaches (Firebase, Serverless computing etc.) is described inside the text, as well as methods of securing the system against cybernetic attacks.

Keywords

Backend-as-a-service; GraphQL; Firebase; Serverless; Flutter

Obsah

1 Požadavky systému

Cílem projektu bylo vytvořit kompletní informační systém pro kulturní festival, který by zprostředkovával komunikaci mezi organizátory festivalu a návštěvníky. Zároveň by měl také přinášet možnost zpětné vazby návštěvníků festivalu k průběhu akce.

1.1.1. Webová administrace

Stěžejní z částí informačního systému je webová administrace. Očekávané funkce webové administrace jsou:

- Jednoduchá a přehledná správa dat, která se objevují v „návštěvnické aplikaci“
- Přehledné zobrazování analytických dat
- Správa uživatelských rolí

1.2. Návštěvnická aplikace

Další částí systému je návštěvnická aplikace, jejíž funkce jsou:

- Přehledné zobrazení line-upu festivalu
- Notifikace uživatelů o změnách v programu
- Interaktivní mapa
- Hodnocení interpretů festivalu
- Registrace a přihlášení uživatelů

2 Používané pojmy

2.1. API

API (Application Programming Interface) označuje rozhraní pro programování aplikací. Jde o sbírku procedur, funkcí, tříd či protokolů nějaké knihovny, které může programátor využívat. API určuje, jakým způsobem jsou funkce knihovny volány ze zdrojového kódu programu. [1]

2.2. GraphQL

GraphQL je dotazovací jazyk pro moderní API, který je přímým konkurentem REST API. [2] Přináší kompletní a pochopitelný popis pro data v API (například typy) a dává možnost dotazovat se jen na data, která opravdu potřebujeme, čímž zamezuje „overfetchingu“. [3]

2.3. Cloud

„Cloud je rozsáhlá síť vzájemně propojených vzdálených serverů po celém světě, které fungují jako jeden ekosystém. Tyto servery jsou navržené buď k ukládání a správě dat, spouštění aplikací, nebo doručování obsahu a služeb, jako je streamování videí, webová pošta, kancelářský software nebo sociální média.“ [4]

2.4. Backend a frontend

Tyto termíny odkazují na oddělení odpovědnosti mezi prezentační vrstvou (frontend) a vrstvou pracující se samotnými daty (backend) softwaru, fyzické infrastruktury a hardwaru. [5] Frontend si tedy můžeme představit jako vše vizuální v aplikacích, zatímco backend jsou části aplikace, které nejsou viditelné pro samotné uživatele a spravují logiku systému.

3 Porovnání Custom Backend vs Backend-as-a-service

Realizaci projektu takového typu, jako je informační systém pro kulturní festival, ovlivňuje volba backendové technologie velmi výrazným způsobem. Bylo proto důležité na začátku vývoje zvolit vhodný typ backendu. V této kapitole se zabýváme jednotlivými možnostmi volby backendu a z ní pak vyplývá proč bylo zvoleno řešení, na kterém je tento informační systém postaven.

3.1. Custom Backend

Jedná se o typ backendu, který je vytvářen od nuly, což poskytuje velikou výhodu ve vytváření backendu přesně na míru danému projektu, ale zároveň je potřeba mnohem více času na vývoj a také udržování samotného backendu v chodu. [6]

3.1.1. Server-side language

Server-side language je typ programovacího jazyka, který je nutný ke komunikaci mezi server, aplikací a databází. Na jeho výběru mohou záviset i výběry databázového nebo aplikačního serveru, proto můžeme považovat Server-side language za hlavní součást backendových technologií. Patří mezi ně např. ASP.NET, Node.js, Python nebo Java. [6]

3.1.2. Database server

Databázové servery jsou backendové programy poskytující databázové služby ostatním počítačům. Používají se k ukládání, hledání a změnám dat a může k nim být přistupováno skrze Server-side language. Mezi nejpoužívanější typy jsou např. Microsoft SQL server, Oracle nebo MySQL. [6]

3.1.3. Application server

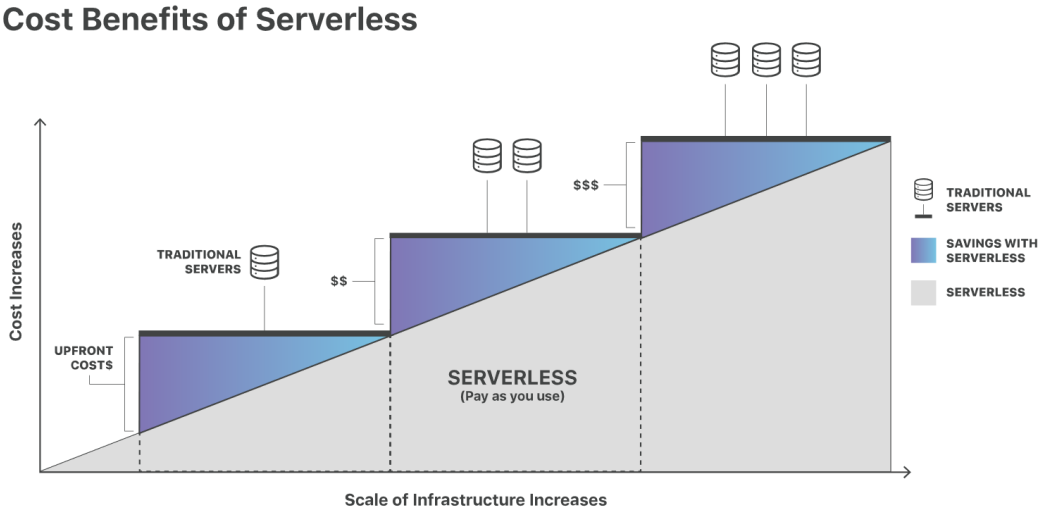
Application server je server jehož účelem je zajišťovat běh aplikací např. webových aplikací. Narozdíl od web serveru, který je určen a optimalizován pro poskytování webových stránek a nemusí mít dostatečný výkon pro spouštění webových aplikací, tak Application server dodává dostatečný výkon a paměť na správnou funkci webových aplikací v reálném čase. [7]

3.2. Backend-as-a-service

Backend-as-a-service je typ cloudové služby, která poskytuje připravené funkcionality, jako jsou třeba databáze, analytika, hosting apod. [8] Dovoluje tak vývojářům starat se jen o vývoj a nezatěžovat se správou serverů. Přináší také jako v případě Serverless computingu, který je podrobně popsán v sekci „Serverless computing“ (3.2.1), alokování zdrojů za běhu služby a jejich přizpůsobení aktuální situaci, čímž zamezuje nedostupnosti aplikace, nebo předejde nadbytečným výdajům.

3.2.1. Serverless computing

Jedná se o typ cloudové nativní architektury, která umožňuje běh backend kódu, kdy za funkčnost serverů a dynamické přiřazování zdrojů odpovídá poskytovatel cloudu. Vývojáři se tak mohou zaměřit pouze na vývoj systému. Výhodami jsou velká škálovatelnost, rychlejší vývoj a snížení nákladů, které můžeme vidět na obrázku níže (Obrázek 1).



Obrázek 1 - Finanční výhody serverless řešení
Obrázek převzat z https://www.cloudflare.com/learning/serverless/what-is-serverless/

3.3. Porovnání backendových řešení

Informační systém pro kulturní festival je specifický především z důvodu nekonzistentního používání systému, jelikož festival probíhá maximálně několikrát do jednoho roka. Proto při rozhodování mezi jednotlivými typy backendových technologií byl kladen důraz převážně na

řešení tohoto specifického požadavku společně s rychlostí vývoje a složitostí jeho udržování, protože za festivalem nestojí týmy vývojářů jako např. v softwarových firmách.

Custom Backend překonává Backend-as-a-service v mnoha ohledech. Jednou z nich může být vývoj backendu přesně na míru danému systému. V backendu tudíž budou jen funkcionality, které jsou opravdu pro systém potřeba. Další nespornou výhodou je udržitelnost backendu do budoucna, protože systém není závislý na službách třetích stran. V případě Backend-as-a-service si nikdy nemůžeme být jisti, zda poskytovatel nezmění ceny služby, jejich funkčnost nebo dokonce jejich řešení nezruší úplně. [9]

Na druhou stranu přináší Backend-as-a-service oproti Custom Backendu rychlejší vývoj a nasazení nebo potřebu zakládat tým backendových vývojářů, kteří by za vývojem backendu a správou serverů, virtuálních strojů nebo containerů stáli. [10] Z logiky věci vyplývá, že toto řešení rovněž přináší snížení nákladů za vývoj backendu a jeho následnou správu. [9] Dále Backend-as-a-service řeší i problém s nekonzistentním používáním systému. Není totiž potřeba pořizovat servery, které běží nepřetržitě i v případě, že nejsou používány, čímž předchází plýtvání zdroji. [11]

Díky svým výhodám byla pro tento informační systém zvolena technologie Backend-as-a-service na platformě Firebase se Serverless API. Běh API zajišťují Firebase Cloud Functions, které jsou jednou z funkcí Backend-as-a-service od Firebase. Více o Firebase Cloud Functions v kapitole „Firebase Cloud Functions“ (5.1.1).

4 Architektura

Z potřeb projektu vyplývá, že se celý informační systém bude skládat ze tří částí. Těmito částmi jsou „webová administrace“ a „návštěvnická aplikace“ o jejichž komunikaci, správu dat a některých funkcí se stará část třetí, a to právě Backend-as-a-service na platformě Firebase.

Při navrhování backendu bylo myšleno na budoucí škálovatelnost samotného systému, ať už z důvodu rozšiřování festivalu nebo integrace i jiných kulturních akcí. Tento problém je vyřešen díky zvolené technologii Serverless API na službě Firebase Cloud Functions, která umožňuje automatický *horizontal scaling*[1]. [12] Pro zrychlení a zjednodušení komunikace API s „webovou administrací“ a „návštěvnickou aplikací“ bylo zvoleno použití dotazovacího jazyka GraphQL podrobně popsaného v sekci „GraphQL“ (2.2).

Jelikož bude systém spravován převážně lidmi, kteří nejsou z oboru IT, byl kladen důraz na co nejjednodušší správu systému. V tomto případě byly zvoleny funkce poskytnuté služnou Firebase, které přináší řadu výhod a ulehčení, jako jsou například správa autentizace a autorizace službou Firebase Authentication, analytická data poskytnutá službou Firebase Analytics nebo služba Firebase Cloud Functions, která poskytuje prostředí pro běh serverless kódu.

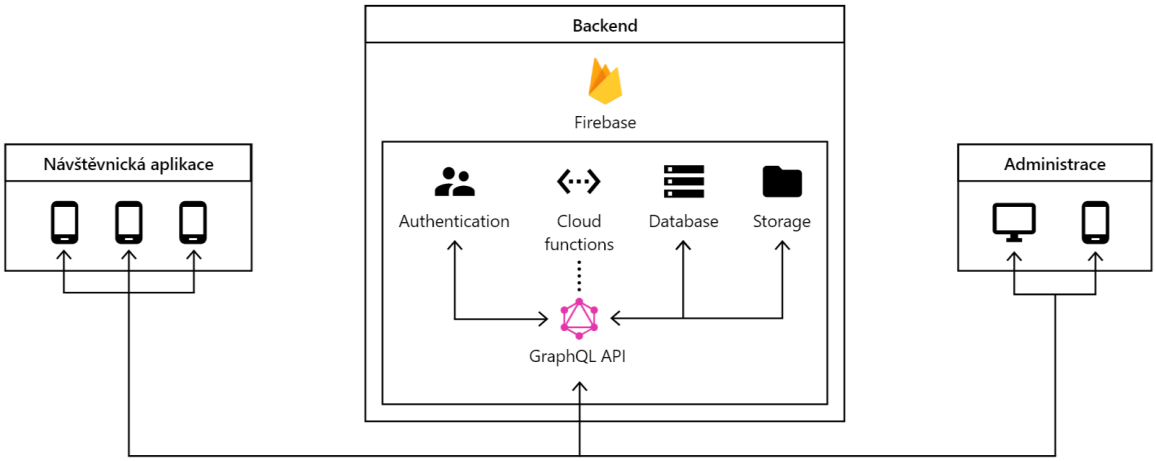
[1] Horizontal scaling – Rozšiřování přidáváním více strojů do systému [28]

Pro tvorbu API bylo nutné zanalyzovat potřeby systému, následně mohla proběhnout samotná tvorba API, proti kterému již byly tvořeny samotné aplikace.

První vytvořenou částí systému byla „webová administrace“, ta poskytuje organizátorům festivalu kontrolu nad daty v „návštěvnické aplikaci“, statistiku o používání nebo správu uživatelských práv.

Druhou částí je „návštěvnická aplikace“, která s využitím dat z databáze poskytuje návštěvníkům všechny důležité informace o festivalu. Jsou to například aktuality, program festivalu nebo notifikuje návštěvníky o změnách v programu.

Tyto části spojené skrze API tvoří celý informační systém pro kulturní festival, nad kterým mají kontrolu organizátoři festivalu skrze „webovou administraci“ a návštěvníci festivalu mohou získávat důležité informace o festivalu skrze mobilní aplikaci. Komunikace mezi jednotlivými částmi systému je vyobrazena na obrázku Obrázek 2 níže.



Obrázek 2 - Návrh architektury systému

5 Backend

Při vývoji informačního systému je nezbytnou součástí vývoje vytvoření funkčního backendu, který se bude starat o zabezpečení, manipulaci s daty, uživatelské role nebo o složitější logiku. Jako platformou pro vývoj backendu byl zvolen Firebase. Celý backend je cloudovou službou typu Backend-as-a-service. K vytvoření Serverless API jsou použity Firebase Cloud Functions.

Následující sekce se zabývá podrobným popisem řešení backendu a v neposlední řadě také některými funkčnostmi, které jsou v backendu tohoto informačního systému nasazeny.

5.1. Firebase

Firebase je platforma na vývoj mobilních a webových aplikací. Jedná se o platformu typu Backend-as-a-service, která poskytuje předpřipravené funkce. Jsou jimi například řešení autentizace skrze Firebase Authentication, databázové služby skrze funkce Firebase Database nebo také storage, kterou zajišťuje funkce Firebase Cloud Storage. Sekce níže obsahují informace o některých z využívaných funkcí Firebase.

5.1.1. Firebase Cloud Functions

Firebase Cloud Functions umožňují automatické spuštění backend kódu, jako odpověď na příchozí požadavek, vyvolaný skrze HTTPS request nebo některou z Firebase služeb. [13] V systému jsou využívány k chodu serverless GraphQL API.

5.1.2. Firebase Cloud Storage

Firebase Cloud Storage umožňuje ukládání a navracení objektů, jako jsou např. fotografie nebo videa. [14] Tento informační systém využívá Cloud Storage k ukládání fotografií jednotlivých interpretů nebo fotografií k některým z aktualit. Cloud Storage rovněž spouští kód ve Firebase Cloud Functions, který automaticky získává a ukládá URL pro stažení daného souboru a ukládá ji do databáze, kde je dále využívána.

5.2. TypeGraphQL

TypeGraphQL je moderní framework pro tvorbu GraphQL API v Node.js. TypeGraphQL přináší do vytváření GraphQL API programovací jazyk TypeScript a s ním rovněž mnoho výhod jako je typový systém nebo dekorátory. Zároveň obsahuje pokročilé funkce např. automatickou validaci, autorizaci skrze dekorátory a mnoho dalších. [15]

5.3. Vybudované backendové řešení

Díky vhodné volbě backendového řešení a technologií je základní funkčnost backendu, jako jsou např. autorizace nebo ukládání objektů do storage vyřešena. Platforma Firebase tyto funkce již poskytuje. To vytváří při vývoji prostor na zaměření se na jiné důležité aspekty systému. Jako jeden z hlavních by zde mohlo být zmíněno zabezpečení systému, kterému byla při vývoji kladena náležitá pozornost.

Jednu z hlavních rolí v backendu informačního systému hraje vhodné API, které komunikuje s jednotlivými částmi systému. Pro tuto úlohu bylo vybráno GraphQL API postavené na frameworku TypeGraphQL. GraphQL jak již bylo zmíněno v kapitole „GraphQL“ (2.2). Jehož největší předností je zamezení *overfetchingu*. Vývojář se tudíž může dotazovat jen a pouze na data, které opravdu potřebuje a nemusí získávat data nadbytečná, jako je tomu např. při použití

REST API. Zároveň podporuje cache stažených dat na zařízeních, tudíž není potřeba získávat data, která už uživatel jednou získal.

5.3.1. Pagination

Pagination na straně API předchází problémům s přenosem velkého množství záznamů, které nemusí být potřeba, to vede ke zpomalení přenosu a zvětšení datového objemu potřebného ke stažení záznamů.

Pagination na straně frontendu se musí zabývat nezredukovanými záznamy. Aplikace se tudíž velice zpomaluje a objem dat potřebných ke stažení záznamů zůstává vysoký. To může vést k problémům především u mobilních zařízení, u kterých se dbá na co nejmenší objem dat potřebných ke stažení záznamů.

V systému je tudíž nasazena pagination v API. Každá query na API, která získává větší množství záznamů z databáze, obsahuje pagination.

5.3.2. Uživatelská oprávnění

Projekt obsahuje 3 skupiny uživatelských rolí, jsou to obyčejní uživatelé, moderátoři a administrátoři. Do webové administrace mají povolený přístup jen moderátoři a administrátoři.

Autentizace probíhá pomocí Firebase Authentication. Pokud autentizace proběhne v pořádku, tak na řadu přijde autorizace, která rovněž probíhá pomocí Firebase Authentication. Každý uživatel nese informaci o své roli v tzv. custom claims.

6 Webová administrace

Webová administrace je webová aplikace pro pořadatele festivalu, přes kterou mají organizátoři přístup ke správě dat, uživatelských rolí apod. Aplikace je přístupná jen pro organizátory festivalu, ti jsou po přístupu do ní schopni spravovat data, která obsahuje „mobilní aplikace“ nebo sledovat analytická data o používání aplikace.

6.1. Funkce aplikace

Organizátoři festivalu mají ve „webové administraci“ možnost spravovat data pro „návštěvnickou aplikaci“. To mimo jiné znamená přidávat, měnit a odstraňovat jednotlivé události festivalu, aktuality apod.

6.1.1. Správa dat

Nejdůležitější funkcí „webové administrace“ je bezesporu správa dat pro „návštěvnickou aplikaci“. Organizátoři festivalu mohou přidávat, měnit nebo odstraňovat jednotlivé události festivalu, aktuality i detailní informace o účinkujících.

6.1.2. Správa uživatelských účtů

Organizátoři festivalu mohou upravovat informace o jednotlivých uživatelských účtech v systému. Jednou z možností je např. úprava jejich uživatelských rolí. Administrátor může měnit role pro jednotlivé uživatele, tím jim přidá, nebo ubere práva, která daný uživatel má.

6.2. Použité technologie

Ve „webové administraci“ jsou použity mimo jiné tyto technologie:

6.2.1. React

React je JavaScriptová knihovna určená k budování uživatelských rozhraní. Může být využit pro budování single-page aplikací nebo mobilních aplikací, protože je vhodný pro práci s rychle se měnícími daty. [16] Základním stavebním blokem React aplikací jsou componentsty jsou renderovány do DOMu pomocí React DOM knihovnou a tím vytváří jednotlivé části uživatelského rozhraní. [17]

6.2.2. Material-UI

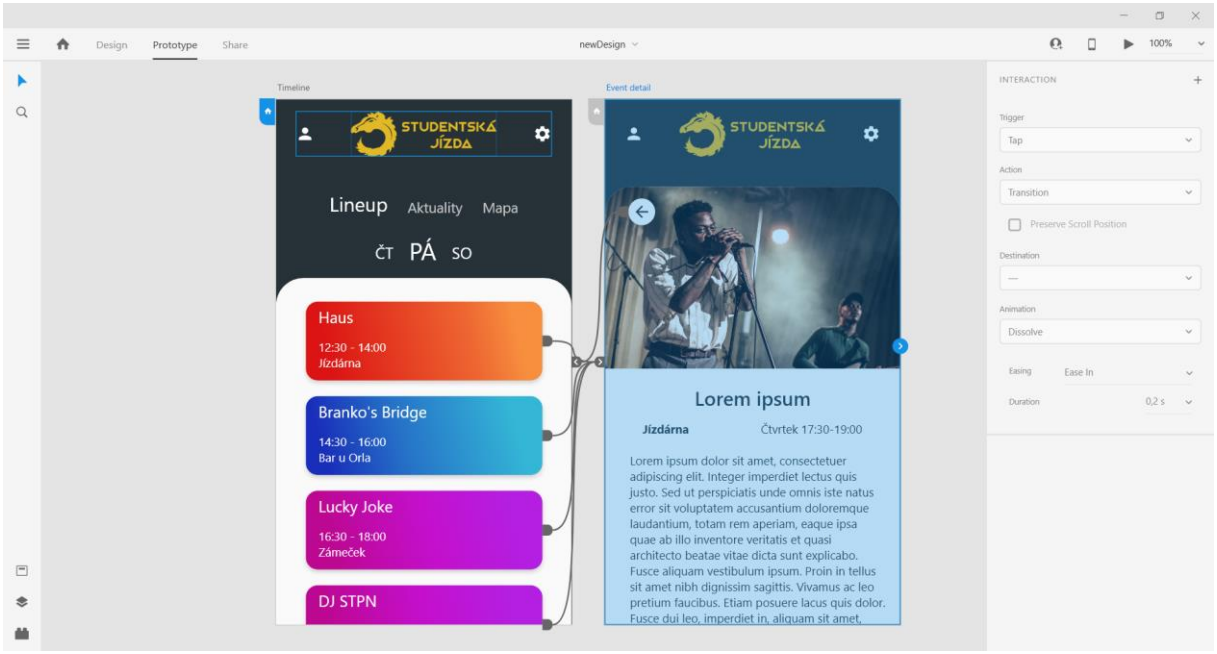
Material-UI je knihovna component pro React, které se dají použít při vývoji webové aplikace a tím ho urychlují. [18] Material-UI vychází ze systému Material design, který je souborem pokynů, komponent a nástrojů, podporující osvědčené postupy tvorby uživatelského rozhraní. [19]

7 Návštěvnická aplikace

Aplikace se zaměřuje na všechny potenciální a současné návštěvníky festivalu. Cílem aplikace je poskytovat návštěvníkům aktuální informace o festivalu a jeho programu nebo jim nabídnout možnost hodnotit jednotlivé interprety.

7.1. Prototyp aplikace

Před začátkem vývoje mobilní aplikace byl vytvořen prototyp samotné aplikace. Ten usměrňoval programování mobilní aplikace tak, aby se netvořily komponenty, které nezapadají do designu aplikace nebo které vůbec nejsou potřeba. Prototyp byl vytvořen na platformě Adobe XD, která podporuje navrhování designu aplikace nebo jeho interaktivní prototyp. Na obrázku Obrázek 3 níže můžeme vidět část prototypu aplikace v programu Adobe XD. Jednotlivé linky mezi komponenty označují přechody a animace v aplikaci.



Obrázek 3 - Prototyp aplikace v programu Adobe XD

7.2. GraphQL client

GraphQL client je kód, který umožňuje posílat POST requesty na GraphQL server. V těle samotného requestu se pak nachází jednotlivé query a mutace, stejně tak i některé proměnné. Po odeslání requestu očekává GraphQL client návrat nějakého JSONu, který na frontendu může být zpracován. [20]

„Návštěvnická aplikace“ využívá jako GraphQL client *graphql_flutter*. Tento GraphQL client zajišťuje „návštěvnické aplikaci“ přenos dat mezi GraphQL serverem a samotnou aplikací.

7.3. Funkce aplikace

Tato sekce se věnuje nejdůležitějším funkcím „návštěvnické aplikace“.

7.3.1. Line-up

Line-up je pravděpodobně hlavní funkcí „návštěvnické aplikace“. Její funkcí je zobrazování aktuálního programu festivalu. Návštěvníci mají tudíž informace o probíhajících akcích, ale také o všech nadcházejících nebo akcích, které již proběhly.

Uživatelské rozhraní line-upu je tvořeno jedním sloupcem ve kterém se nacházejí kartičky jednotlivých událostí festivalu. Pokud bude uživatel scrollovat v line-upu směrem dolů, pak GraphQL client automaticky získá další data, protože počet současně získaných záznamů na API limituje pagination (viz sekci 5.3.1).

7.3.2. Interaktivní mapa

Interaktivní mapa zobrazuje mapu kulturního festivalu a aktuální pozici návštěvníka. Pokud se tedy uživatel aplikace na festivalu ztratí, nebo bude potřebovat najít jiné pódium, může k navigaci na správné místo využít tuto interaktivní mapu.

7.3.3. Notifikace

Jednou z dalších funkcí „návštěvnické aplikace“ je zobrazování notifikací, které informují o změnách v programu nebo mimořádnostech, jako jsou např. špatně zaparkované auto nebo ztracená peněženka.

7.3.4. Aktuality

„Návštěvnická aplikace“ může být využívána k zobrazení článků sepsaných organizátory festivalu. Podobné články slouží k propagování festivalu, informování o účinkujících v dalších ročnících, zobrazování novinek o festivalu apod.

7.4. Použité technologie

Při volbě technologií pro vývoj mobilní „návštěvnické aplikace“ bylo cílem vybrat takové technologie, které by splňovaly požadavky na multiplatformní vývoj, rychlost aplikace a zároveň nebyly příliš časově náročné na vývoj. V této sekci se zaměřujeme na volbu vhodné technologie pro „návštěvnickou aplikaci“.

7.4.1. Xamarin

Xamarin je open-source platforma pro tvorbu multiplatformních aplikací. Je postaven na .NET Frameworku a pro vývoj je použit jazyk C#. [21] Zásadní výhodou tohoto řešení je, jak již bylo zmíněno výše, postavení vývoje na frameworku .NET, který má za sebou velkou komunitu vývojářů a podporu ze strany Microsoftu. Xamarin rovněž přináší téměř nativní výkon aplikací nebo vývoj grafického rozhraní podobného tomu nativnímu. [22]

7.4.2. Flutter

Flutter je open-source UI SDK pro vývoj multiplatformních aplikací v jazyce Dart. [23] Poskytuje velké množství výhod, jako jsou např. hot reloady[2], katalog widgetů[3] nebo dokonce výkon totožný s nativními aplikacemi díky kompilování jazyka Dart do nativního jazyka ARM zařízení. [24]

[2] Hot reload – obnovení souboru ve kterém byla provedena změna, bez ztráty stavu aplikace [29]
[3] Widget – základní stavební bloky UI ve Flutteru, udržují popis o svém vzhledu, jejich konfiguraci a stavu [30]

7.4.3. Porovnání technologií

Přestože jsou obě technologie výhodné pro vývoj multiplatformních aplikací, tak v případě aplikace pro kulturní festival se Flutter jeví jako lepší volba vzhledem k rychlosti vývoje a následně i lepšího výkonu aplikace. Po stránce designu je Flutter opět vítězem, ačkoli Xamarin podporuje nativní design, náročnost dosažení kvalitního vzhledu je několikanásobně složitější než v případě Flutteru. [22]

8 Zabezpečení systému

Každý IT systém musí myslet na možná bezpečnostní rizika a u online systému to platí několikanásobně Tato sekce se zaměřuje na zabezpečení celého informačního systému, způsoby, jakými se předchází nejčastějším útokům a také některým vybraným bezpečnostním prvkům vytvořeného systému.

8.1. Autorizace query a mutací na API

Mutace a query odesílané na API určené jen pro administrátory a moderátory musí být vhodně autorizovány. Základní uživatel nesmí mít práva ke smazání záznamů z databáze. Naopak administrátor takové změny provést může.

Autorizace na straně API probíhá pomocí *JWT* (JSON web token). Tokeny jsou generovány službou Firebase Authentication a při každém zabezpečeném requestu je ověřováno, zda má daný uživatel právo na vykonání akce, zda tokenu již nevypršela platnost a zda nebyl token upraven. Na základě tohoto ověření pak API vykoná danou akci, nebo vrátí chybové hlášení.

8.2. Příklady útoků

Níže jsou popsány některé z nejběžnějších kybernetických útoků dle měření OWASP. [25] Autor si je vědom těchto rizik vědom a výsledný systém je na takovéto útoky připraven.

8.2.1. NoSQL injection

Jedná se o typ útoku, při kterém jsou využívány neošetřené uživatelské vstupy. Útočník tím může získat autorizovaná data, modifikovat obsah databáze nebo zničit celý systém. Na obrázku Obrázek 4 si můžeme všimnout jednoduchého NoSQL injection útoku, který by mohl získat jména a emailové adresy uživatelů. [26]

```
{
    users(search: "{\"email\": {\"$gte\": \"\"}}",
          options: "{\"fields\": {}}") {
        _id
        username
        fullname
        email
    }
}
```

Obrázek 4 - Příklad NoSQL injection útoku
Obrázek převzat z http://www.petecorey.com/blog/2017/06/12/graphql-nosql-injection-through-json-types/

Prevencí tohoto útoku je kontrola samotných uživatelských vstupů a jejich validování, ke kterému dochází v API.

V tomto informačním systému je útoku typu injection zabráněno správnou volbou frameworku pro tvorbu API. TypeGraphQL totiž používá typový systém, tudíž do parametrů jednotlivých

dotazů nemůže být vloženo nic neočekávaného. Na obrázku Obrázek 5 můžeme vidět jednoduchý Input type, který je jasnou definicí vstupního objektu parametru.

```
@InputType()
class EventInput implements Partial<Event>{

    @Field()
    title: string;

    @Field()
    place: string;

    @Field()
    image: string;

    @Field(type => Datetime)
    end: Date;

    @Field(type => Datetime)
    start: Date;

    @Field()
    detail: string;
```

Obrázek 5 - Náhled jednoduchého input typu

Pokud je takovýto Input type použit v parametru query nebo mutace, jak je vidět na obrázku Obrázek 6, tak při requestu na API není možné do parametru vložit nic jiného, jako je tomu například u GraphQL scalar typu JSON.

```
@Authorized(["MODERATOR", "ADMIN"])
@Mutation(() => Event)
async addEvent(
  @Arg("event")
  { title, detail, end, start, image, place }: EventInput
) {
  const docRef = await admin
    .firestore()
    .collection("events")
    .add({
      title,
      detail,
      end,
      start,
      image,
      place
    });
  const data = (await docRef.get()).data();
  const uid = (await docRef.get()).id;
  return { ...data, uid };
}
```

Obrázek 6 - Náhled použití InputType v GraphQL mutaci

8.2.2. Cross-site scripting

Cross-site scripting je metoda útoku na webové aplikace, která využívá především neošetřené vstupy a díky tomu dokáže do stránky vkládat svůj vlastní škodlivý kód, který by mohl vést ke změně vzhledu, odstavení aplikace, či krádeži dat. [27]

Tomuto typu útoku lze zabránit např. escapováním znaků na vstupech. Díky volbě Reactu jako knihovny pro vývoj „webové administrace“ tomuto útoku předcházíme. React veškeré vstupy tzv. escapuje, čímž zamezuje cross-site scripting útokům, při kterých je například do textových polí vložen škodlivý JavaScriptový kód uvnitř HTML tagu. Pokud by se takovýto text vypsal bez escapování, tak by to mohlo vést k jednomu z již výše zmíněných negativních následků.

9 Závěr

Byl vytvořen informační systém pro organizátory a návštěvníky kulturního festivalu. „Webová administrace“, která umožňuje správu dat, uživatelů nebo zasílání notifikací o nastalých mimořádnostech. „Návštěvnická aplikace“ umožňující návštěvníkům festivalu náhled line-upu, detailní informace o účinkujících, zobrazování notifikací nebo interaktivní mapu.

9.1. Budoucí možnosti práce

Do budoucnosti tohoto informačního systému je plánováno rozšíření i pro jiné kulturní akce. Ze systému by se tak mohla stát platforma pro libovolné kulturní akce v České republice, ale i jinde ve světě.

10 Zdroje a odkazy

[1] „Wikipedia,“ 18 Říjen 2019. [Online]. Available: https://cs.wikipedia.org/wiki/API. [Přístup získán 3. Březen 2020].

[2] Nautron, „GraphQL,“ [Online]. Available: http://graphql.cz/. [Přístup získán 2. Březen 2020].

[3] GraphQL Foundation, „GraphQL,“ [Online]. Available: https://graphql.org/. [Přístup získán 2. Březen 2020].

[4] MICROSOFT CORPORATION, „Microsoft Azure,“ 2020. [Online]. Available: https://azure.microsoft.com/cs-cz/overview/what-is-the-cloud/. [Přístup získán 11. Březen 2020].

[5] „Wikipedia,“ 31. Prosince 2018. [Online]. Available: https://cs.wikipedia.org/wiki/Front_end_a_back_end. [Přístup získán 11. Březen 2020].

[6] M. Techlabs, „Medium,“ 26 Červenec 2016. [Online]. Available: https://medium.com/@MarutiTech/how-to-choose-the-perfect-back-end-technology-5674db9c0192. [Přístup získán 5 Březen 2020].

[7] Sharpened Productions, „Tech terms,“ 22. Listopad 2019. [Online]. Available: https://techterms.com/definition/application_server. [Přístup získán 6. Březen 2020].

[8] M. Májský, „Zdroják,“ 8. Srpen 2016. [Online]. Available: https://www.zdrojak.cz/clanky/zaciname-s-backend-service/. [Přístup získán 3. Březen 2020].

[9] S. Merenych, 20. Srpen 2019. [Online]. Available: https://clockwise.software/blog/choice-between-mobile-backend-as-a-service-and-custom-backend/. [Přístup získán 6. Březen 2020].

[10] Cloudflare, Inc., „Cloudflare,“ 2020. [Online]. Available: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/. [Přístup získán 6. Březen 2020].

[11] „Cloudflare,“ 2020. [Online]. Available: https://www.cloudflare.com/learning/serverless/why-use-serverless/. [Přístup získán 4 3 2020].

[12] Firebase, Inc., „Firebase,“ 3 Prosinec 2019. [Online]. Available: https://firebase.google.com/docs/functions. [Přístup získán 6. Březen 2020].

[13] Google, Inc., „Google Cloud,“ [Online]. Available: https://cloud.google.com/functions/. [Přístup získán 18. Leden 2020].

[14] Google, Inc., „Firebase,“ 28. Leden 2020. [Online]. Available: https://firebase.google.com/docs/storage. [Přístup získán 9. Březen 2020].

[15] M. Lytek, „TypeGraphQL,“ [Online]. Available: https://typegraphql.ml/. [Přístup získán 9. Březen 2020].

[16] „Wikipedia,“ 4. Říjen 2020. [Online]. Available: https://cs.wikipedia.org/wiki/React_(webov%C3%BD_framework). [Přístup získán 10. Březen 2020].

[17] „Wikipedia,“ 8. Březen 2020. [Online]. Available: https://en.wikipedia.org/wiki/React_(web_framework). [Přístup získán 10. Březen 2020].

[18] Material-UI, „Material-UI,“ 2020. [Online]. Available: https://material-ui.com/. [Přístup získán 10. Březen 2020].

[19] Google, Inc., „Material design,“ [Online]. Available: https://material.io/. [Přístup získán 10. Březen 2020].

[20] A. Aiyer, „Medium,“ 17. Únor 2018. [Online]. Available: https://medium.com/open-graphql/exploring-different-graphql-clients-d1bc69de305f. [Přístup získán 11. Březen 2020].

[21] MICROSOFT CORPORATION, „Microsoft,“ 16 Září 2019. [Online]. Available: https://docs.microsoft.com/cs-cz/xamarin/get-started/what-is-xamarin. [Přístup získán 7. Březen 2020].

[22] N. Sharma, „Apptunix,“ 17. Září 2018. [Online]. Available: https://www.apptunix.com/blog/frameworks-cross-platform-mobile-app-development/. [Přístup získán 7. Březen 2020].

[23] „Wikipedia,“ 2. Březen 2020. [Online]. Available: https://en.wikipedia.org/wiki/Flutter_(software). [Přístup získán 7. Březen 2020].

[24] Google, Inc., „Flutter,“ [Online]. Available: https://flutter.dev/?gclid=Cj0KCQiAqY3zBRDQARIsAJeCVxNq4ZgRFurM-

ATLK0MQFBbdCynag9VW_SmCY2RtsYLqsAEmz42mBD4aAo-MEALw_wcB. [Přístup získán 7. Březen 2020].

[25] OWASP Foundation, Inc., „OWASP,“ 6. Únor 2020. [Online]. Available: https://owasp.org/www-project-top-ten/. [Přístup získán 6. Březen 2020].

[26] P. Corey, „Pete Corey,“ [Online]. Available: http://www.petecorey.com/blog/2017/07/03/what-is-nosql-injection/. [Přístup získán 9. Březen 2020].

[27] „Wikipedia,“ 4. Únor 2020. [Online]. Available: https://cs.wikipedia.org/wiki/Cross-site_scripting. [Přístup získán 3. Březen 2020].

[28] A. Korpal, „Medium,“ 15. Leden 2019. [Online]. Available: https://medium.com/@abhinavkorpal/scaling-horizontally-and-vertically-for-databases-a2aef778610c. [Přístup získán 6. Březen 2020].

[29] N. Dabit, „Stackoverflow,“ 2. Leden 2017. [Online]. Available: https://stackoverflow.com/questions/41428954/what-is-the-difference-between-hot-reloading-and-live-reloading-in-react-native. [Přístup získán 7. Březen 2020].

[30] Google Inc., „Flutter,“ [Online]. Available: https://flutter.dev/docs/development/ui/widgets-intro. [Přístup získán 7. Březen 2020].

11 Seznam obrázků