VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SPRÁVA VÝSTAVNÍCH PLOCH PRO FESTIVAL

MANAGEMENT OF EXHIBITION SITES FOR A FESTIVAL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **RADEK DUCHOŇ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. ZBYNĚK KŘIVKA, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Duchoň Radek**
Program: Informační technologie
Název: **Správa výstavních ploch pro festival**
Management of Exhibition Sites for a Festival
Kategorie: Informační systémy

Zadání:

1. Seznamte se s procesy pro správu výstavních ploch a vystavovatelů na festivalu Animefest. Seznamte se také s vhodnými webovými technologiemi (např. Laravel, Symfony či Django) a případně i prostorovými databázemi (např. PostGIS).
2. Dle konzultací s vedoucím proveďte návrh informačního systému, který bude pomáhat se správou výše zmíněných procesů i přehledným zobrazováním využití spravovaných ploch. Bude-li to vhodné využijte při návrhu prostorovou databázi.
3. Návrh implementujte jako webovou aplikaci, která bude mít vhodnou část také responzivní, aby ji bylo možné ovládat i z dotykového mobilního zařízení. Implementuje také přehledné grafické zobrazení ploch včetně jejich rozmístění a dodatečného zobrazení souvisejících informací.
4. Systém uživatelsky testujte alespoň s 10 uživateli, pak jej nasaďte do provozu a jeho nasazení zhodnoťte a navrhněte případný další vývoj.

Literatura:

- Garrard, C.: Geoprocessing with Python. Manning Publications, 2016, 360 s.
- Obe, R.O., Hsu, L.S.: PostGIS in Action. 2. vydání, Manning Publications, 2015, 600 s.
- Pavelek, M.: Podpora pořádání festivalových soutěží. FIT VUT v Brně, 2017. Bakalářská práce.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a část bodu 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Křivka Zbyněk, Ing., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 31. července 2020
Datum schválení: 21. října 2019

Abstrakt

Cílem této práce je vytvoření nového informačního systému pro správu výstavních ploch na festivalu Animefest. Webová aplikace je implementována v jazyce Python za využití aplikačního rámce Django. Systém využívá pro definici a reprezentaci prostor vektorovou mapu ve formátu SVG. Vytvořené řešení poskytuje také vlastní systém pro záznam historie změn v databázi. Systém umožní organizátorům festivalu katalogizovat prostory a uchovávat historii spolupráce s různými partnery.

Abstract

The aim of this work is to create a new information system for the management of exhibition spaces at the festival called Animefest. The web application is implemented in Python using the Django application framework. The system uses a vector map in SVG format for the definition and representation of spaces. The created solution also provides its own system for recording the history of changes in the database. The system will allow festival organizers to catalog spaces and preserve the history of cooperation with various partners.

Klíčová slova

festival, výstava, webová aplikace, informační systém, uživatelské rozhraní, Django, Python 3, PostgreSQL, HTML, CSS, JavaScript, jQuery, Bootstrap, MVC, JSON, XML, SVG

Keywords

festival, exhibition, web application, information system, user interface, Django, Python 3, PostgreSQL, HTML, CSS, JavaScript, jQuery, Bootstrap, MVC, JSON, XML, SVG

Citace

DUCHOŇ, Radek. *Správa výstavních ploch pro festival.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zbyněk Křivka, Ph.D.

Správa výstavních ploch pro festival

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Zbyňka Křivky. Další informace mi poskytli pan Adam Rambousek a paní Kristýna Mikešová. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Radek Duchoň
31. července 2020

Poděkování

Tímto bych chtěl poděkovat panu Ing. Zbyňku Křivkovi, PhD. za odborné vedení při této bakalářské práci. Dále bych chtěl poděkovat svému spolužáku Josefu Oškerovi, který mi poskytl server pro testování a dále poskytoval technickou pomoc při problémech se serverem.

Obsah

Kapitola 1

Úvod

Cílem této bakalářské práce je vytvoření nového informačního systému pro správu výstavních ploch na festivalu zvaném Animefest. Partnerům bude systém sloužit k rezervaci výstavních ploch, nábytku, služeb a vstupenek pro dobu konání festivalu. Organizátorům bude naproti tomu pomáhat jednak se správou a katalogizací prostor a také s kontrolou historie úprav důležitých údajů u jednotlivých partnerů a jejich objednávek.

Animefest začal jako relativně malé setkání pár stovek fanoušků japonského komiksu (mangy) a animovaného filmu (anime) pořádaný zapsaným spolkem Brněnští otaku. Po letech pořádání však narostl a stal se největším festivalem v Česku a na Slovensku, kam každoročně jezdí tisíce lidí.[2] S tím také musely narůst prostory, které vyplňují mimo jiné i různorodé stánky s tematickým zbožím, občerstvením, nebo se zde nachází také třeba herní koutky s arkádovými a deskovými hrami.

S rozšiřováním festivalu narůstá i obtížnost vhodného rozvržení prostor a přehledného udržování informací o jednotlivých partnerech, proto je vhodné navrhnout a vytvořit nový informační systém s dostatečně detailní a flexibilní databází. Analýzou požadavků pro zajištění použitelnosti se zabývá kapitola 2, kde budou rozebráni typy uživatelů a jaké jsou jejich požadavky na výsledný systém, tedy co v něm budou moct dělat.

Po analýze požadavků je nutno vybrat technologie vhodné pro implementaci a vypracovat návrh systému, který bude zjištěné informace o nárocích na informační systém reflektovat. Pro implementaci byl zvolen jazyk Python 3, který se těší stále větší oblibě mezi uživateli a aplikační rámec Django, který nad tímto jazykem pracuje a slouží k vývoji webových aplikací. Tyto technologie jsou doplněny databázovým systémem PostgreSQL, které nabízí nejvíce možností při práci s rámcem Django. Detailnější zdůvodnění jejich výběru a popis dalších používaných technologií a lze nalézt v první části kapitoly 3 a dále v rozmezí kapitoly 4, kde se nachází také například vysvětlení návrhových vzorů MVC a MVT.

Mezi nejdůležitější požadavky při návrhu webové aplikace vyvíjené v rámci této práce patří například dobrý způsob zaznamenávání změn v databázi informačního systému, primárně za účelem možnosti zpětného dohledávání chyb zanesených uživateli systému. S tím se také neodmyslitelně pojí návrh dostatečně detailní databáze. Tomuto návrhu systému se tato práce věnuje v kapitole 3.

Po návrhu aplikace a vybrání implementačních technologií bylo možno přemístit se k řešení samotné implementace, které je věnována praktičtěji zaměřená kapitola 5. V rámci této kapitoly lze vidět také ukázky z uživatelského rozhraní aplikace.

Závěrečná kapitola 6 pak shrnuje způsob, kterým bylo prováděno testování aplikace. Následuje závěrečné shrnutí práce s návrhy pro budoucí rozšíření aplikace.

Kapitola 2

Analýza požadavků

Nejprve byla provedena analýza současného stavu, aktuální systém funguje na principu vygenerování unikátní URL (Uniform Resource Locator) adresy po odeslání registračního formuláře. Z této adresy lze upravovat a doplňovat další údaje, partner tak může upřesňovat objednávku a např. přidávat dodatečný nábytek. Detailnější požadavky se dále vyřizují skrze emailovou a telefonickou komunikaci. To je možné až dokud správce nepotvrdí registraci s výslednou cenou. Na základě požadavků všech partnerů se správce prostor snaží vymyslet vhodné uspořádání subjektů k co největší spokojenosti všech.

Požadavky na webovou aplikaci byly vypracovány primárně na základě konzultací s vedoucím práce a dalšími organizátory Animefestu. Do systému se budou moci přihlásit organizátoři akce a partneři, kteří zde budou moci registrovat své subjekty. Systém pak musí být lokalizovaný do češtiny a angličtiny s možností rozšíření o další jazyky v budoucnu.

2.1 Uživatelské role a vazby

Konkrétní zpřístupněné funkce informačního systému se budou odvíjet od uživatelských skupin rolí a vazeb mezi subjekty a uživateli. Přihlašování uživatele probíhá pomocí uživatelského jména, nebo emailu a hesla, případně za pomoci autentizace skrz třetí strany za využití protokolu OAuth2. Systém bude obsahovat celkem 5 rolí – uživatel, zodpovědná osoba, správce sekcí, správce prostor a administrátor. Jeden uživatel může mít i více rolí.

Uživatel může být po žádosti (například pomocí emailu) povýšen do role zodpovědné osoby, což mu umožní registrovat subjekty a přiřazovat další uživatele k subjektům, za které je zodpovědný. Těmto uživatelům bude moci eskalovat práva, čímž získají možnost též přiřazovat další uživatele, spravovat údaje subjektu a pomáhat s rezervacemi pro daný subjekt. Subjekty budou řazeny do různých sekcí, které mají své správce.

Organizátoři festivalu se budou rozdělovat na správce sekcí, správce prostor a administrátora, přičemž administrátor bude mít všechna existující oprávnění. Správci sekcí pak mají za účel primárně rozhodovat o přijmutí, či zamítnutí registrací subjektů v sekci, na kterou se specializují. Správce prostor následně hlavně vyřizuje rezervace vstupenek a spravuje žádosti o rezervaci vybavení, služeb a prostor pro dříve registrované subjekty. Zároveň bude mít možnost zadávat do systému nové zóny s plochami pro rezervaci jednotlivými subjekty a určovat cenu jednotlivých ploch. Zóny bude moct členit do podsekcí, dle kterých se bude určovat, kterými sekcemi subjektů jsou v nich obsažené plochy rezervovatelné.

Administrátor, správce prostor i správce sekcí mohou měnit údaje subjektů a případně také prohlížet zaznamenané logy se změnami v databázi.

Administrátor bude jediný, kdo bude mít přímý přístup do administračního rozhraní poskytovaného aplikačním rámcem Django, navrch může také upravovat ceny a slevy, zadávat pokuty na akci, spravovat uživatele, kterým může libovolně přiřazovat role. Omezeně však smějí spravovat uživatele také správci prostor a sekcí, a to hlavně pro možnost přiřazení role zodpovědné osoby uživatelům, s jinými rolemi však manipulovat nemohou.

2.2 Partneři a subjekty

Každý subjekt bude registrován zodpovědnou osobou neboli partnerem do určité sekce (např. stánek s občerstvením), která může být zohledňována při výpočtu cen a vymezuje možnou lokalitu a maximální kapacitu prostor, které si lze pronajmout.

Kvůli účetnictví bude muset subjekt obsahovat mimo kontaktních informací také fakturační údaje, kolik má být celkově zaplaceno a kolik již bylo z této částky uhrazeno.

K subjektu poté půjde přiřadit další osoby, které s ním budou moci v omezené míře pracovat, toto oprávnění půjde eskalovat označením osoby za správce, po povýšení bude moct daný uživatel přidávat k subjektu další osoby (nemůže je však označit za správce) a účastnit se rezervací pro daný subjekt.

Zodpovědná osoba a případné další pověřené osoby s eskalovanými právy budou podávat žádosti k rezervaci prostor, nábytku, služeb a objednávat vstupenky.

Ke každému subjektu bude přiřazeno určité množství vstupenek, v základu se odvíjející od rezervovaných prostor. U vstupenek bude možno zadat, zda jsou na určité jméno a pokud, tak na jaké, případně přiřadit jednoduše některého uživatele z osob přiřazených k subjektu a využívat údaje rovnou z profilu. Vstupenky by mělo jít hromadně exportovat např. do formátu CSV.

2.3 Zaznamenávání změn

Vzhledem k velikosti akce narostl počet lidí, kteří musí se systémem pracovat, s tím se také zvýšila šance chyby lidského faktoru. Z toho důvodu je nutné dělat záznamy o jednotlivých akcích, aby bylo zpětně možné snáze dohledat zdroj takových omylů a tyto chyby následně eliminovat.

Výpis změn by mělo být možné filtrovat podle řady parametrů – například dle uživatele, který je provedl, subjektu, ke kterému se vážou, nebo časového rozmezí, ve kterém byly změny provedeny.

Mělo by také být možné přihlásit se k odběru emailů s informací o zaznamenávaných změnách v určitých databázových modelech.

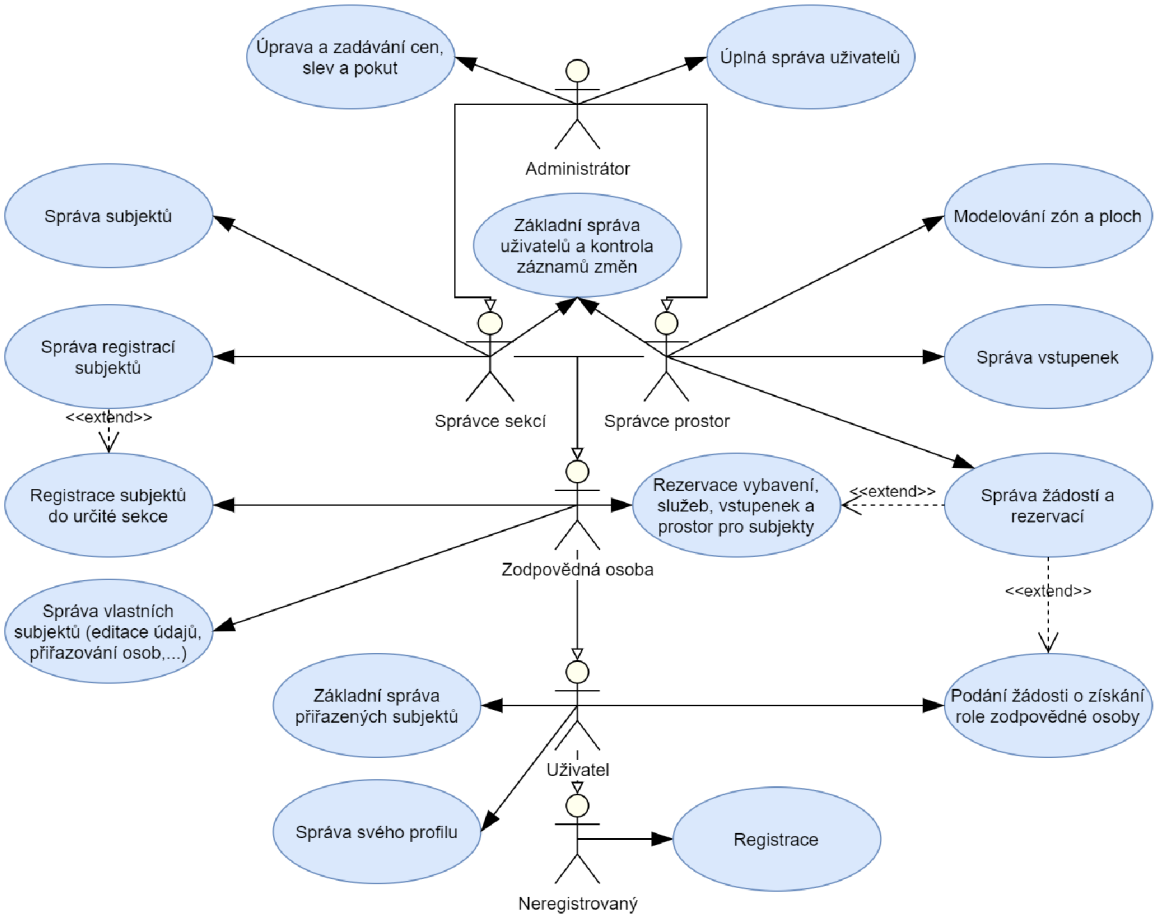
Kapitola 3

Návrh informačního systému

V této kapitole je popsáno, v jakých technologiích je informační systém vyvíjen a proč jsem se pro dané technologie rozhodl. Následuje návrh databáze a systému pro uchovávání historie změn v databázi aplikace.

Bude se tedy jednat o webovou aplikaci s hlavním zaměřením funkčnosti na desktopová zařízení, s omezenou funkčností by však měla být podporována také mobilní zařízení.

Na níže zobrazeném diagramu případů užití lze vidět typy uživatelů v systému a funkce, které pro ně bude nutno implementovat.



Obrázek 3.1: Diagram případů užití.

3.1 Použité technologie pro server

Tato podkapitola bude věnována počátečnímu výběru technologií pro vývoj serverové části webové aplikace, jmenovitě tedy aplikačního rámce a databázového systému.

Aplikační rámec

V dnešní době je na výběr z poměrně velkého množství aplikačních frameworků pro webové aplikace a není tak problém vybrat si i například podle toho, pro který programovací jazyk má programátor své preference. Nabízí se například Laravel a Symphony pro jazyk PHP, Django a Flask pro Python, Spring pro vývoj v programovacím jazyce Java, nebo třeba ASP.NET pracující nad jazykem C#.

Současně patří mezi nejrozšířeněji používané Django a Laravel, které se v řadě žebříčků vyskytují takřka zpravidla mezi nejlepšími deseti, díky čemuž pro ně existuje široká škála návodů a rozšiřujících knihoven.

Django patří mezi starší aplikační rámce, poprvé vyšlo v roce 2005. Stáří nicméně není problém díky neustálé údržbě a vývoji firmou Django Software Foundation. Za tu dobu navíc vzniklo takřka bezkonkurenční množství návodů, které jsou v naprosté většině případů kompatibilní i s novějšími verzemi tohoto aplikačního rámce. V opačném případě je v podstatě zpravidla integrována nějaká novinka, která funkci ve starším návodu nahrazuje, přičemž nalézt náhradu vyřazených funkcionalit není nikterak obtížné. Aplikační rámec Laravel je ve srovnání s rámcem Django novější, pochází z roku 2011. Nejspíše hlavně z tohoto důvodu je dle statistik[1] Laravel použit v méně projektech a má méně přispěvovatelů, z toho lze usuzovat, že Django bude mít z pohledu kvantity návodů navrch. Zároveň je Django oproti rámci Laravel citelně rychlejší, což je dáno hlavně rychlostí programovacího jazyka Python ve srovnání s jazykem PHP. V kombinaci s tím, že Animefest, pro který je systém vyvíjen, již má k dispozici jeden informační systém implementovaný v rámci Django, rozhodl jsem se taktéž pro tento aplikační framework.

Databázový systém

Po výběru aplikačního rámce se významně omezil výběr databázového systému, pro který již není nikterak velký výběr. Django aktuálně, jak se zmiňuje dokumentace, podporuje tyto 4 systémy[4]: PostgreSQL, MySQL, Oracle a SQLite.

Případně lze využít databázové systémy, ke kterým bylo naprogramováno rozhraní třetími stranami, těmi jsou IBM DB2, Microsoft SQL Server, FireBird a ODBC. U těch se však liší specifické schopnosti databázových dotazů dle podpory poskytnuté třetími stranami, řada návodů by tak nejspíše nešla využít a ani vlastnosti popsané v dokumentaci rámce Django nemusí odpovídat.

SQLite je svobodné velmi minimalistické řešení databáze, které je relativně efektivní v rychlosti, nezabere mnoho místa a je snadno přenositelné. Django jej podporuje ve verzi 3.8.3 a novější. SQLite je však pro svůj minimalismus v některých funkcích a datových typech omezené, například nemá formát pro přesné vyjádření čísel s plovoucí řádovou čárkou, kde by tedy docházelo automaticky k zaokrouhlování.

Oracle je uzavřený relační databázový systém, který je podporován ve verzi 12.1 a vyšší a pro svou funkci s Djangem vyžaduje cx_Oracle Python ovladač. Narozdíl od ostatních podporovaných databází nepodporuje Oracle databáze rozlišení hodnoty `NULL` od prázdného řetězce, pro něž používá stejnou reprezentaci a jeho textová pole jsou do jisté míry limitována v indexaci, funkcích a délce.

MySQL je otevřený relační databázový systém. Django jej podporuje ve verzi 5.6 a vyšší, pro jeho funkci vyžaduje rámec Django navíc doplněk mysqlclient. Omezení ve srovnání se dříve zmíněnými variantami nejsou tak velká, jedná se o relativní drobnosti. Příkladem může být, že unikátní pole znaků může mít maximální délku 255 znaků, textovým polím bez specifikace horního limitu pak nelze nastavit unikátnost, nebo nefungující indexy pro efektivní řazení modelů. Jsou však známy implementační chyby, které přetrvávají i v aktuálně nejnovějších verzích, kdy se za určitých podmínek může například změnit jméno databázové tabulky. To by mohlo způsobit problémy se záznamem změn, jehož návrh bude řešen v podkapitole 3.3.

PostgreSQL je svobodný a otevřený relačně-objektový databázový model. Spolu s Djangem jej lze využít ve verzi 9.4 a vyšší, pro svou funkčnost vyžaduje adaptér psycopg2[18]. Netrpí žádnými ze zde dříve zmíněných omezení ostatních databází, podporuje naopak navíc některá specifická pole, jako například pole pro JSON, nad rámec specifikace v dokumentaci aplikačního rámce Django. Dále podporuje navíc možnost zvýšení izolační úrovně transakcí a lze dále rozšířit pomocí doplňků, jako jsou např. PostGIS pro geografické modely a hstore pro hashovou mapu, nabízí tak větší možnosti pro budoucí rozšíření aplikace. Z důvodů výše zmíněných nedostatků ostatních databázových systémů a v tomto odstavci zmíněných pozitiv PostgreSQL bylo zvoleno pro implementaci právě PostgreSQL.

3.2 Návrh databáze

Aplikace musí uchovávat ke svému provozu poměrně velké množství dat pro velké množství entit. V této podkapitole budou vysvětleny důležité prvky databáze, na které byl kladen důraz při návrhu. Entity a vztahy mezi nimi jsou dále také graficky znázorněny v přiloženém entitně-vztahovém diagramu (angl. Entity-Relationship Diagram) 3.2.

3.2.1 Uživatel a třídy poskytované rámcem Django

Základ uživatelské části databáze staví na třídách, které poskytuje samotný rámec Django, těmi jsou `User`, `Group` a `Permission`. User má jednak unikátní neměnitelné ID, ale také unikátní uživatelské jméno, dále obsahuje údaje jako jméno, příjmení, informace o přidělených oprávněních, časové údaje o registraci a posledním přihlášení, položku udávající, jestli je účet aktivní, tj. jestli se k němu dá přihlásit, kvůli velkému množství vazeb je totiž vhodnější účty pouze deaktivovat než úplně mazat. Hodnota `Staff` říká, jestli se lze přihlásit do administrátorského rozhraní a hodnota `Superuser` dává uživateli implicitně všechna existující oprávnění.

Uživateli lze přiřazovat oprávnění, kromě vlastních specifikovaných oprávnění jsou pro každou třídu databázového modelu generovaná čtyři oprávnění implicitně – zobrazit, přidat, změnit a smazat. Tato implicitně generovaná oprávnění mají mimo jiné efekt na možné operace s instancemi jednotlivých tříd v administrátorském rozhraní. Pro potřeby používání velkého počtu oprávnění, nebo úprav oprávnění pro velké množství uživatelů najednou, lze tato oprávnění přiřadit i konkrétním skupinám a do těchto skupin připojit jednotlivé uživatele.[5]

Třída `User` je navíc rozšířá vlastní třídou `Profile`. Profil uživateli přidává další údaje pro telefonní číslo a označení, zda již uživatel ověřil svůj email. Dále pak nahrazuje email uživatele, který je sice obsažen i ve třídě User, nicméně není unikátní a bylo by tedy nepřiměřeně náročné dodatečně provádět potřebné kontroly explicitně.

3.2.2 Partneři a sekce subjektů

Partnery jsou míněny jednotlivé subjekty, kterými jsou například prodejní stánky, nebo kreslíři. Jedna zodpovědná osoba tak může registrovat více subjektů. Každý subjekt má vlastní kontaktní a fakturační údaje, které se nemusí shodovat s údaji zodpovědné osoby. Mezi fakturačními údaji se vyskytuje také DIČ, u kterého probíhá kontrola prostřednictvím systému VIES[3] (VAT Information Exchange System) pro potřeby DPH ve členských státech EU. Dále obsahuje informace určené ke sdílení pro návštěvníky, místo pro poznámky partnera a organizátorů, údaje o souhrnné ceně za poskytnuté služby a o jejich zaplacení. Na závěr v sobě objekt nese informaci o tom, zda tato žádost k registraci subjektu čeká na vyřízení, či zda již byla schválena nebo zamítnuta, v případě schválení bude následovat ještě stav uzavření objednávek, po čemž již nebude možno vytvářet nové objednávky.

Subjekty jsou registrovány do konkrétních sekcí, které obsahují omezení, do kdy lze subjekty v dané sekci registrovat, maximální velikost rezervovatelných prostor. Sekce mají vliv na výpočet cen, lze tak nastavit slevu na jednotlivé kusy nábytku pro určité skupiny.

Ke každému subjektu lze přiřadit další osoby jako pracovníky, těm pak lze nastavit vyšší práva pro práci se subjektem pomocí nastavení na správce subjektu. Zodpovědná osoba je správcem implicitně a může přiřazené osoby správci jmenovat, tyto osoby však další dalším uživatelům práva eskalovat nemohou.

Partnerovi lze přiřazovat vstupenky, které mohou a nemusí být na určitá jména, v případě nastavení konkrétního uživatele se bude automaticky používat jméno z jeho profilu. Během akce lze partnerům udělovat i pokuty.

3.2.3 Nábytek, vybavení a služby

Partner si může objednat od organizátorů předem specifikované služby. Objednávka služeb obsahuje stav informující, zda se čeká na vyřízení, je určena k realizaci, či byla třeba zamítnuta. Obdobně je vymyšlený nábytek, který si může partner objednat z předem specifikovaných objektů od organizátorů Animefestu. Vše musí být schváleno správcem prostor nebo administrátorem.

Nábytek je pojmenovaný a nese informace o ceně a osobě, která model do databáze vložila, objednávky obsahují informace o tom, kdo je kdy vytvořil.

3.2.4 Zóny, oblasti a rezervace ploch

Partner si může zarezervovat prostory, přičemž nejvýše si lze zarezervovat prostory dle předem specifikovaného maxima velikosti prostor daného sekcí partnera. Prostory se budou řadit do zón, v rámci kterých bude specifikované, které sekce partnerů si v nich mohou prostory rezervovat.

Zóna bude reprezentována pomocí vektorové mapy, v níž budou mít rezervovatelné objekty identifikační klíč, který bude unikátní v rámci zóny. Za pomoci klíčem opatřených objektů v mapě se budou v databázi generovat oblasti, kterým bude možno přiřadit různé ceny.

Oblasti budou přidělovány do podsekcí zóny. tyto podsekce budou definovat, které sekce partnerů si mohou rezervovat která místa. Oblast tedy bude rezervovatelná za předpokladu, že se nachází v alespoň jedné podsekci zóny, která povoluje rezervace pro danou sekci subjektu.

Rezervace prostor bude probíhat výběrem oblastí v definované vektorové mapě. Prostory budou označené jako rezervované ihned po vložení žádosti o rezervaci oblastí, uvolní

se, pakliže bude žádost o rezervaci prostor zamítnutá. V případě schválení rezervace se vygenerují volné již schválené vstupenky pro partnera za každou rezervovanou oblast.

3.3 Záznamy změn v databázi

Pro možnost jednoduchého zaznamenávání téměř všech změn napříč databází je nutné navrhnout vlastní třídu. Jelikož zaznamenávat vždy celý stav v podobě hluboké kopie by bylo vysoce neefektivní pro náročnost na úložiště a přehledné zobrazení historie změn objektu by touto metodou bylo vysoce náročné, rozhodl jsem se pro přístup, kdy jsou ve třídě uloženy údaje, které lze využít pro získání konkrétní instance libovolné třídy z databáze. Tento přístup dovoluje zaznamenávat pouze měněné údaje, zároveň lze relativně snadno filtrovat záznamy změn dle konkrétní třídy databázového modelu nebo jeho instance.

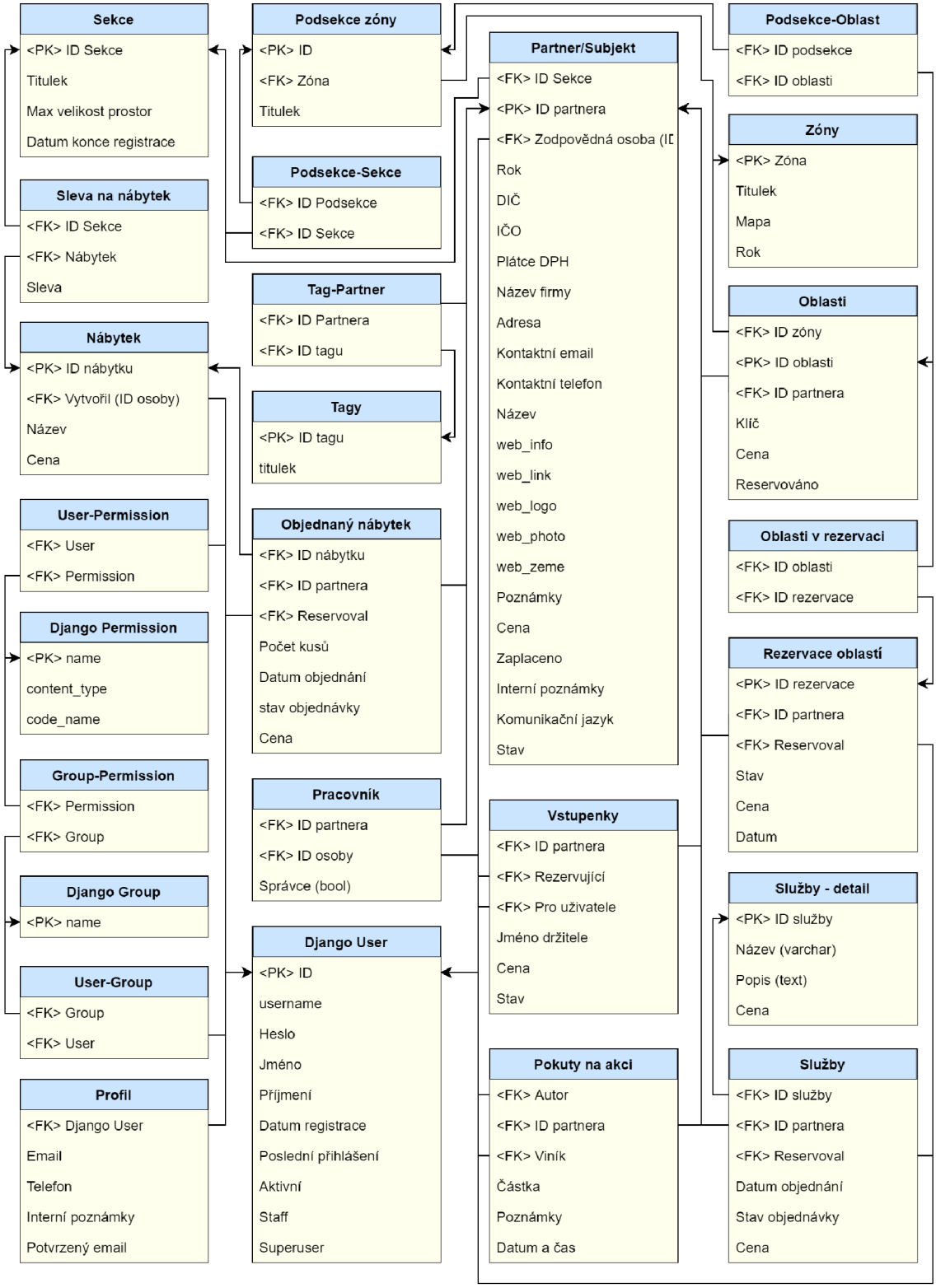
Ze zaznamenaných údajů půjde rozlišit, jaký byl stav objektu před změnou a po změně. Dále budou zaznamenávány informace říkající, kdo změny provedl a kdy je provedl, ze záznamů však budou vynechány některé specifické údaje, jako je například změna hesla. Aby se předešlo chybám, bude nutné, aby objekty nebyly z databáze plně mazány, pokud k nim již existuje nějaký záznam o změně, případně pokud už, tak by se měly smazat i příslušné logy. Odstranění příslušných logů by však bylo náročnější, jelikož se zde nevyskytují takzvané cizí klíče (angl. foreign key) a nelze tedy jednoduše nastavit kaskádové mazání, záznamy by tedy bylo nutné odstranit explicitními příkazy. Vhodnější varianta je proto například pouze nastavit údaj vyjadřující neplatnost objednávky či neaktivitu uživatele, což zaručí i zachování historie změn.

Dále bude třída automaticky zasílat oznámení o změnách emailem na adresy, které budou mít přihlášený odběr, který půjde nastavit buď pro konkrétní databázové třídy, nebo pro všechny zaznamenávané změny. Třída se bude zároveň starat o vhodnou reprezentaci dat. Místo názvu obrázku nebo cizích klíčů odkazujících na instance jiných tříd databázového modelu by se tak mohla vracet například URL adresa tak, aby bylo možné v pohledu stránky otevřít odkaz směřující na detail daného objektu.

3.4 Internacionalizace a lokalizace

V aplikaci je nutné vyřešit problém lokalizace do českého a anglického jazyka. Aplikační rámec Django poskytuje vestavěné funkce za účelem plné podpory při internacionalizaci a lokalizaci vyvíjených projektů, kde internacionalizací se myslí příprava aplikace pro překlad prováděná programátorem, lokalizací je myšlen překlad a lokální formátování, tedy akce prováděné hlavně překladatelem[7].

V základu Django umožňuje specifikovat, které části textů se mají překládat a do kterých jazyků se mají překládat, přičemž velká část textů v základních knihovnách rámce Django je již přeložená do mnoha jazyků, mezi nimi i právě do češtiny. Tyto překlady lze snadno upravit či rozšířit a stejně tak lze rozšířit i samotný výběr jazyků, přičemž dokud není aplikaci dodán překlad daného řetězce, bude používán v původní podobě. To usnadňuje možnost lokalizace v rámci jednoho jazyka, např. v případě chtěné lokalizace mezi britskou a americkou angličtinou může být zapotřebí upravit jen malou část výrazů.



Obrázek 3.2: ER diagram.

Primární klíče, které nebyly nijak potřebné pro modelování vazeb mezi entitami, nejsou v diagramu výše vyobrazeny. Není-li v definici třídy uvedeno jinak, aplikační rámec Django automaticky generuje pro každý databázový model identifikátor v podobě autoinkrementačního celého čísla (angl. integer).

Kapitola 4

Použité technologie

V této kapitole jsou popsány technologie využívané v implementaci a případně informace o využívaných verzích a jejich zjištěných vzájemných omezeních.

4.1 Serverová část (Back-end)

Serverovou část (angl. Back-end) lze obvykle rozdělit na tři části, těmi jsou server, aplikace a databáze. Běžný uživatel má k těmto částem přístup pouze nepřímo a do značné míry omezený. Tento přístup je zprostředkovaný pomocí klientské části.

Python

Python[16] je univerzální interpretovaný vysokoúrovňový jazyk, který je dostupný v mnoha verzích, které by ve většině případů měly být zpětně kompatibilní. V tomto jazyce je napsána serverová část aplikace a popis databáze.

Pro Python 2 je od tohoto roku (2020) oficiálně ukončená podpora, pro tuto práci je tedy využit Python 3. Konkrétně je práce vyvíjena ve verzi Pythonu 3.7.5, novější verze 3.8.0 sice byla dostupná již z počátku vývoje, avšak nebyla kompatibilní s webovým aplikačním rámcem Django v poslední dosud vydané verzi s dlouhodobou podporou.

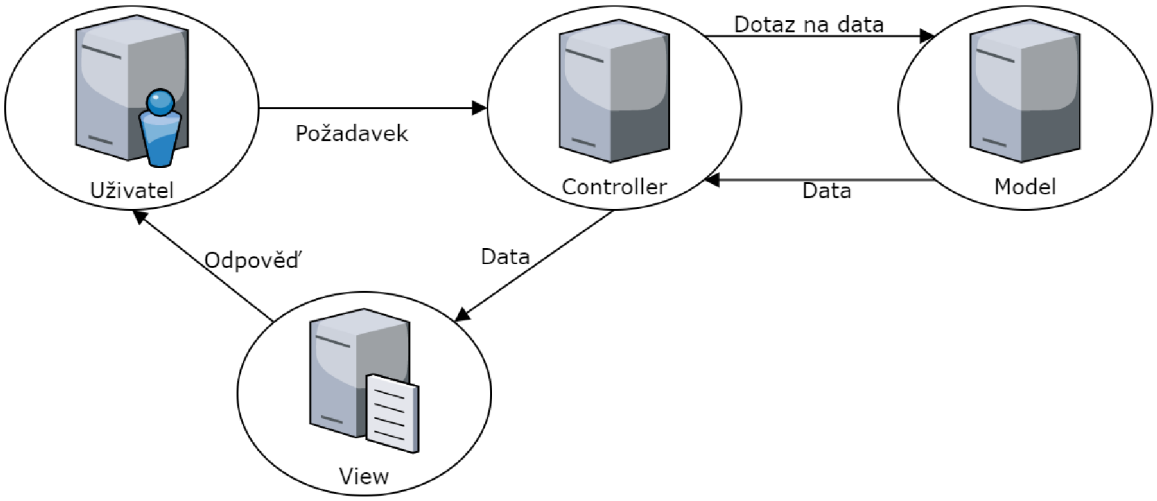
PostgreSQL

PostgreSQL[17] je objektově-relační databázový systém, který zde slouží pro trvalé uchování údajů o uživatelích a partnerech na serveru. Jedná se o open source (otevřený) software, na jehož vývoji se podílí globální komunita vývojářů, díky čemuž pro něj existuje řada doplňků. Mimo to obsahuje i v základu některá specifická pole nad rámec standardu běžných databázových systémů, příkladem může být pole pro ukládání formátu JSON.

Django

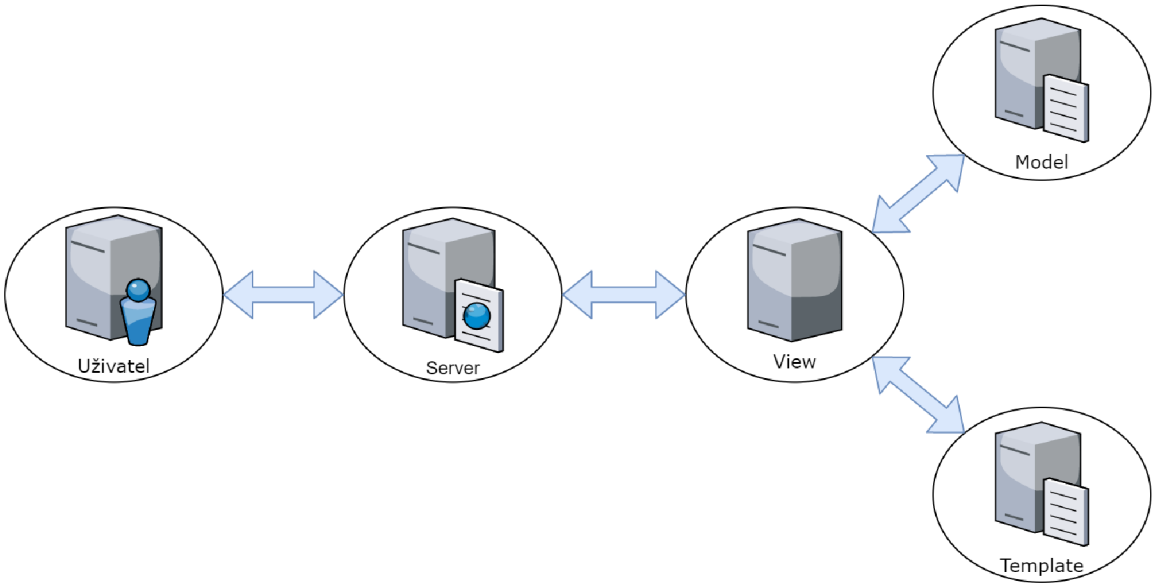
Django[5] je na programovacím jazyku Pythonu založený webový aplikační rámec, jenž se drží architektury MVT (Model-View-Template), která vychází z principu architektury MVC (Model-view-controller). Tyto architektury oddělují datové modely, uživatelské rozhraní a řídicí logiku do tří nezávislých komponent.

Schémata těchto architektur jsou znázorněna v přiložených diagramech 4.1 a 4.2.



Obrázek 4.1: Schéma MVC architektury.

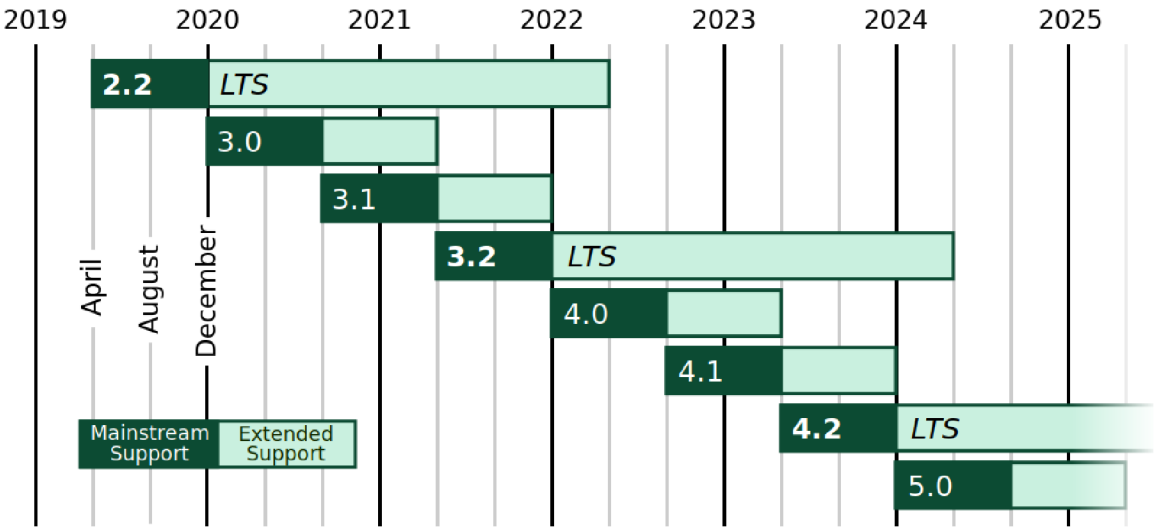
Model v MVC architektuře obstarává správu dat, logiku a omezení aplikace. Pohled (angl. View) má za účel zpracování dat pro jejich zobrazení a poskytuje různé reprezentační komponenty. Pojem „view“ zde nemá souvislost s týmž pojmem, který se vyskytuje také v rámci Django. Kontroler (angl. Controller) manipuluje s modely a rendruje pohledy, funguje tak jako most mezi modelem a pohledem[9].



Obrázek 4.2: Schéma MVT architektury.

Model v MVT architektuře slouží jako rozhraní pro data, jedná se ve své podstatě o databázi. Pohledy obsahují byznys logiku, pomocí níž manipulují s modely a vykreslují šablony (angl. template rendering), přijímají HTTP požadavek (angl. HTTP request) a vrací HTTP odpověď (angl. HTTP response). Šablona je komponenta, která slouží jako prezentační vrstva, v základu se jedná o HTML kód pro rendrování dat. Asi největším rozdílem ve srovnání s MVC architekturou je mapování URL adres, které provádí server pro spuštění příslušných pohledů, tento prvek se v MVC architektuře nevyskytuje[9].

Jak lze vidět z diagramu 4.3 níže, jediná podporovaná verze rámce Django v době začátku vývoje byla verze 2.2, což je zároveň verze s přislíbenou dlouhodobou údržbou. To znamená, že bezpečnostní záplaty budou dle plánů vycházet ještě alespoň do začátku roku 2022. Zmíněná verze je z těchto důvodů používána v tomto projektu.



Obrázek 4.3: Diagram znázorňující vývoj a plánovanou podporu verzí[8].

4.2 Klientská část (Front-end)

Klientská část (angl. front-end) je interaktivní část aplikace přístupná uživatelům. Tato část aplikace je implementována za pomoci webových šablon, které poskytuje Django a kde se dále kombinují běžně užívané jazyky – HTML, CSS a JavaScript, kde HTML a CSS slouží primárně pro statické zobrazení obsahu, JavaScript pak pro dynamické zobrazení a akce na straně klienta.

HTML

HTML[14] (Hyper Text Markup Language) je značkovací jazyk, který je určen pro popis WWW (World Wide Web) stránek za pomoci tzv. značek (angl. tag) a jejich atributů, pomocí nichž definujeme hlavně sémantiku dokumentu. (Např. značka `<p>` označuje odstavec. Některé značky jako třeba `<span>` však sémantický význam nemají.) Dříve se dokumenty pomocí atributů i stylizovaly, ale od toho se již upustilo, standardně se dnes dokumenty stylizují pomocí tzv. kaskádových stylů.

CSS

CSS[13] neboli „Kaskádové styly“ (angl. Cascading Style Sheets) je jazyk pro popis způsobu zobrazení jednotlivých prvků v HTML struktuře, stejně tak i vzhledu celého dokumentu pomocí pravidel. Pravidla se skládají ze selektoru (popis značek, pro které budou styly platit) a z bloků deklarací (popis chtěných vlastností) oddělených středníky. Styly lze vpisovat přímo do HTML značek, ale lze je i oddělit, a to buď jinam do dokumentu (typicky do

hlavičky) jako takzvané stylopisy (angl. stylesheets), nebo do externích souborů, kterým se v takovém případě přezdívá externí stylopisy.

JavaScript

JavaScript[15] je vysokoúrovňový objektově orientovaný skriptovací jazyk, jehož běhové prostředí je řízené událostmi a běží ve webovém prohlížeči na straně klienta.

Využívána je zde rozšiřující JavaScriptová otevřená a svobodná knihovna jQuery[10], která do značné míry usnadňuje práci s JavaScriptem, který se snaží podobně jako kaskádové styly oddělit od struktury HTML kódu za pomoci definic selektorů. Selektory určují, se kterými prvky stránky má skript interagovat, není tak potřeba přidávat kód pro spuštění událostí přímo do atributů HTML značek.

Dále je využíván otevřený a svobodný JavaScriptový rámec jQuery UI, který implementuje sadu interakcí a efektů pro uživatelské rozhraní na základě dříve zmíněné JavaScriptvé knihovny jQuery[11]. Z tohoto rámce je využívaná hlavně třída `Selectable`, která umožňuje snadné označení a výběr prvků stránky, přičemž lze snadno definovat, které prvky mají být vybratelné za pomoci jQuery selektorů. Také je možné například definovat vlastní akce, které se mají spustit po výběru prvků, či zrušení jejich výběru.

JavaScript je v aplikaci využíván například pro funkce jako „označit vše“, vynucení potvrzovacího okna před provedením změn, ale i asynchronní dotazy na server pro předběžnou kontrolu formuláře, či obdržení konkrétních dat bez nutnosti znovu načíst celou stránku. Asynchronními dotazy se snižuje nárazová zátěž serverové části aplikace, jelikož je díky rozdělení problémů na menší podproblémy v jednu chvíli nutno jen menších operací. Není tak zapotřebí např. po každé operaci vykreslovat celou stránku. Klesá ale i náročnost na přenos dat mezi uživatelem a serverem, jelikož se hlavně mnohdy relativně objemný základ stránky nemusí přenášet mezi serverem a klientem vždy.

Bootstrap

Bootstrap[12] je volně dostupná sada nástrojů pro vývoj a design webů a webových aplikací.

V podobě knihoven rozšiřuje CSS o šablony, pomocí nichž poskytuje základní definice stylů pro běžné HTML prvky za účelem sjednocení jejich designu a poskytnutí základní responzivity. Mimo jiné jsou zde obsaženy také třídy pro tvoření modálních oken.

Dále Bootstrap obsahuje JavaScript knihovnu, která využívá dříve zmíněnou jQuery knihovnu a přidává další rozšíření dostupných funkcí a objektů. Bootstrap knihovnu je proto nutné vložit do dokumentu až po jQuery knihovně.

Kapitola 5

Implementace

V této kapitole bude věnován prostor samotné implementaci výše zmíněných návrhů a požadavků. Nejdříve bude popsána implementace serverové části, která byla vytvořena v programovacím jazyce Python 3. Posléze bude následovat popis implementace klientské části, pro kterou je využit Django šablonovací jazyk ve spojení se značkovacím jazykem HTML, kaskádovými styly a skriptovacím jazykem pod názvem JavaScript.

5.1 Serverová část

Tato podkapitola se bude zabývat popisem implementace serverové části aplikace. Jako první se bude věnovat databázi webové aplikace, následovat bude sekce věnující se oprávněním uživatelů a skupin, dále pak budou probrány pohledy, internacionalizace projektu a záznamník změn.

5.1.1 Databáze

Schéma databáze prošlo od původního návrhu v podkapitole 3.2 během průběhu implementace mnoha změnami. Definice všech vlastních modelů databáze probíhá v modulu `models.py` prostřednictvím tříd. V databázi se však nachází ještě další modely, které se využívají. Jedním takovým databázovým modelem je třída User, na kterou se váže mnoho funkcí aplikačního rámce Django, které souvisejí například s autentizací (viz sekce 5.1.2), dokonce se na tuto třídu váží některé balíčky, např. balíček `social-auth-app-django` přidává do databáze modely, které slouží pro přiřazení účtů např. sociálních sítí, jako je Facebook či Twitter, ale i Google účet k uživateli za využití OAuth2 autentizace skrze API (Application Programming Interface) daných služeb.

Příklad definice vlastního databázového modelu pomocí třídy lze vidět v ukázce zdrojového kódu 5.1. Lze zde vidět, jak probíhá odkazování se na jinou tabulku pomocí tzv. cizího klíče (angl. foreign key), což v praxi znamená, že se zde ukládá primární klíč z instance odkazované třídy databázového modelu. V těchto případech je nutno definovat, co se má stát, kdyby byl referovaný řádek tabulky smazán, v tomto případě se neodehraje nic, ale v jiných případech mohou být například smazány všechny závislé entity. Reference přitom musí mít pevně určený databázový model, na který je odkazováno. Dále si lze povšimnout hodnoty pro datum, tam je při zadaném parametru `auto_now_add=True` automaticky ukládán aktuální den a čas při vzniknutí záznamu. Podobným způsobem by šla nastavit i například nějaká základní konstantní hodnota, nebo hodnota proměnné, čehož v některých případech též využívám, takové hodnoty jsou však zapsány v jednom konfiguračním souboru – modulu

`settings.py`. Oproti tomu například u uživatele nelze bohužel zařídit tak snadno, aby se automaticky ukládal do záznamu nyní přihlášený uživatel, který vytvoření záznamu inicioval. Dále každá třída obsahuje implicitně jednu položku, kterou není potřeba explicitně definovat, tou je primární klíč. Pokud není uvedeno jinak, tak je primární klíč celé číslo, které se při vytvoření nového záznamu automaticky inkrementuje o 1, čímž se stává unikátním v rámci třídy. Tento automatický systém umožňuje mimo jiné snadno ukládat kopie již dříve vytvořených instancí, pro tyto potřeby stačí vymazat z instance hodnotu primárního klíče, při pokusu o uložení instance se následně uloží duplikát s novou hodnotou primárního klíče.

Výpis 5.1: Třída pro záznamy změn databáze

```
1  from django.db import models
2  from django.contrib.auth.models import User
3
4  class Log(models.Model):
5      user = models.ForeignKey(User, on_delete=models.DO_NOTHING)
6      date = models.DateTimeField(auto_now_add=True)
7      data = JSONField()
8      app_label = models.TextField()
9      table_name = models.TextField()
10     field_pk = models.TextField()
```

5.1.2 Uživatelské skupiny a oprávnění

Django má vlastní systém skupin oprávnění, které lze udělovat jednak konkrétním uživatelům a jednak lze také přidělit oprávnění přímo celé skupině uživatelů. Uživatel má oprávnění potud, pokud jej má přidělené jako jednotlivec nebo má ono oprávnění alespoň jedna ze skupin, jichž je členem. Implicitně jsou generována čtyři oprávnění pro každý databázový model, a to oprávnění pro přidání nových instancí a zobrazení, změnu a smazání stávajících instancí daných databázových tříd. Administrátor má implicitně všechna oprávnění, a to i za předpokladu, že nejsou v rámci aplikace vůbec definovaná. Při vývoji je tedy třeba dávat pozor, aby bylo oprávnění uvedeno přesně a neobsahovalo syntaktickou chybu, pokud je jím totiž něco podmíněno v šabloně, administrátor nebude mít příležitost zaznamenat problém s chybným uvedením. Žádný jiný uživatel nicméně dané oprávnění mít ani nemůže, pokud je v něm nějaký překlep, může tak snadno vzniknout chyba. Při vývoji je proto vhodné testovat aplikaci i s účtem bez práv administrátora.

V implementace je pak zařízeno, aby každý nově vytvořený uživatel byl implicitně zařazen do skupiny `user`, což umožňuje snadno v případě potřeby všem uživatelům udělit nějaké oprávnění a později jej zase odebrat, když už nebude zapotřebí. Tohoto implicitního zařazení bylo docíleno za využití dekorátoru typu `receiver`, který přijímá signál po uložení instance třídy User, pokud byl uložením vložen nový záznam do databáze, je uživateli následně přidělena skupina, přičemž je pro něj vytvořen i rozšiřující profil, jak lze vidět na ukázce kódu 5.2 dále.

Název oprávnění	Využití oprávnění v uživatelském rozhraní
give_responsible_person	Přiřazování role zodpovědné osoby
view_admin_panel	Zobrazení administračního panelu
view_zone_details	Procházení aktuálního stavu zón a podsekcí
change_permission	Přiřazování libovolných rolí uživatelům
view_permission	Zobrazení oprávnění uživatelů
change_logs	Dočasné přepnutí zhotovování záznamů změn
view_logs	Zobrazení záznamů změn
add_stall	Registrace subjektů
change_stall	Změna údajů a vyřizování žádostí subjektů
change_profile	Změna údajů uživatelů
view_profile	Zobrazení profilů uživatelů
add_stallfurniture	Objednání nábytku pro subjekty
change_stallfurniture	Změna objednávek nábytku subjektů
view_stallfurniture	Zobrazení objednávek nábytku subjektů
add_stallticket	Objednání vstupenek pro subjekty
change_stallticket	Změna objednávek vstupenek subjektů
view_stallticket	Zobrazení objednávek vstupenek subjektů
add_stallservice	Objednání služeb pro subjekty
change_stallservice	Změna objednávek služeb subjektů
view_stallservice	Zobrazení objednávek služeb subjektů
add_stallpenalty	Udělování pokut subjektům
change_stallpenalty	Změna pokuty subjektů
view_stallpenalty	Zobrazení udělených pokut
add_areareservation	Rezervace prostor pro subjekty
change_areareservation	Změna rezervací prostor
view_areareservation	Zobrazení rezervací prostor
add_zone	Definice nových zón a podsekcí s prostorami
change_zone	Správa definovaných zón a podsekcí
delete_zone	Zrušení definovaných zón a podsekcí

Tabulka 5.1: Seznam využívaných oprávnění

Výpis 5.2: Třída pro záznamy změn databáze

```
1 from django.contrib.auth.models import User, Group
2 from django.db.models.signals import post_save
3 from django.dispatch import receiver
4
5 @receiver(post_save, sender=User)
6 def update_profile_signal(sender, instance, created, **kwargs):
7     if created:
8         Profile.objects.create(user=instance)
9         instance.groups.add(Group.objects.get_or_create(name='user')[0])
```

Jak již bylo dříve zmíněno, pro každý databázový model jsou implicitně generována čtyři oprávnění, která definují, které operace může uživatel s danými modely provádět přímo v administračním rozhraní poskytovaném aplikačním rámcem Django, přičemž se jedná o práva zobrazení, přidání, změnu a smazání instancí konkrétních databázových tříd. I v rámci implementace jsem pracoval primárně s těmito implicitními oprávněními, jelikož se nepředpokládá běžný přístup uživatelů k administrátorskému rozhraní informačního systému. Výjimkou jsou tři oprávnění, která jsou vygenerována explicitně, všechna tři jsou přiřazena ke třídě `Profile`. První z nich umožňuje zobrazení administračního panelu v rámci uživatelského rozhraní. Odtud by měli správci prostor a sekcí vykonávat své akce vyžadující vyšší práva, jako je správa objednávek či uživatelů. Druhým je oprávnění pro přiřazení role `zodpovědná osoba` uživatelům v pohledu správy uživatelů, důvodem pro toto oprávnění je, aby správci prostor a sekcí měli možnost přiřadit pouze tuto jednu roli. Role správců však smí registrovaným uživatelům přidělit pouze administrátor. Posledním explicitním oprávněním je právo na zobrazení pohledu pro procházení zón a jejich podsekcí. Tento pohled slouží primárně pro účely rychlých kontrol během akce, mělo by se tak jít rychle například prokliknout k údajům subjektu na základě lokace.

Všechna využívaná oprávnění včetně jejich funkce v uživatelském rozhraní lze vidět v tabulce 5.1 výše. Lze si povšimnout, že s výjimkou prostor nejsou využívána oprávnění pro zrušení instancí databázových tříd, odstraňování záznamů totiž není vhodné pro účely uchování změn v databázi. Výjimkou v úplném odstraňování jsou objednávky subjektů, dokud je objednávka ve stavu, kdy nezačala být vyřizována, mohou ji správci subjektu smazat. Tato možnost byla implementována hlavně pro možnost rychlé nápravy chyby, kdyby bylo například omylem objednáno větší množství nábytku, než bylo chtěné.

5.1.3 Autentizace uživatelů

Django poskytuje základní funkce pro autentizaci uživatelů, jelikož však email uživatelů není standardně v rámci Django unikátní, bylo nutné pro možnost přihlašování pomoci emailu vytvořit vlastní pohled a funkci. Funkce se v takovém případě pokusí získat uživatele pomocí unikátního emailu v profilu uživatele a toho následně teprve zkouší standardně autentizovat.

Mimo základní autentizace je v aplikaci zapojená knihovna `social-auth-app-django`, ta umožňuje přihlášení a registraci uživatelů pomocí účtu u aplikací třetích stran. Tato metoda funguje pomocí protokolu OAuth2, díky kterému není potřebné sdělovat své přihlašovací údaje aplikaci třetí strany, skrz kterou se uživatel snaží přihlásit. Pro fungování této autentizace je standardně nutné vytvořit si účet, vyplnit údaje a splnit podmínky, které si třetí strana klade. Sociální síť Facebook tak například vyžaduje šifrované připojení ke stránce pomocí protokolu HTTPS (Hypertext Transfer Protocol Secure). Twitter a společnost Google takové šifrování nevyžadují, vyžadují však uvedení stránek s podmínkami služeb (angl. terms of service) a zásadami ochrany osobních údajů (angl. privacy policy). Po vyplnění všech požadovaných údajů je uživateli svěřená dvojice údajů - tzv. ID klienta a tajný klíč klienta. Tyto údaje je nutné zadat do globálního konfiguračního souboru projektu `settings.py`, aby autentizace skrze třetí stranu fungovala.

V případě využití aplikace třetí strany pro autentizaci je nejprve zkontrolováno, zda již existuje k danému účtu vazba, pokud ano, účet, ke kterému se vztahuje, je přihlášen. V opačném případě se pokusí vyhledat účet se stejnou emailovou adresou, jako má účet u aplikace třetí strany, ten se v případě nalezení spáruje, v případě nenalezení vhodného účtu bude za pomoci údajů získaných z aplikace třetí strany vytvořen nový účet. Přihlá-

šený uživatel navíc může spárování s účty třetích stran kdykoliv zrušit či autorizovat účet z dalších podporovaných aplikací.

5.1.4 Pohledy

Pohledy rámce Django (angl. Views) jsou funkce, které přijímají webové požadavky a vrací webové odpovědi, ty mohou mít podobu HTML obsahu pro stránku, obrázku, nebo čehokoliv jiného. Pohled samotný obsahuje veškerou nutnou logiku pro vrácení odpovědi [6].

Výběr zavolané pohledové funkce je řízen za pomoci webové URL adresy. Mapování URL adres pro konkrétní funkce probíhá v modulech `urls.py`, kterých může být i více v různých podsložkách. První modul, který je bezpodmínečný, se nachází ve složce projektu s globálním nastavením. Zde se mapují na různé prefixy další vzory adres, přičemž například mapování pro administrátorskou sekci poskytuje samotný rámec Django, užití lze vidět v ukázce 5.3. Obdobně fungují také některé balíčky, v tomto případě balíček pro autentizaci pomocí protokolu OAuth2. Dále zde lze vidět namapovaný modul `urls.py`, který se nachází ve složce `accounts`. V tomto modulu se nachází vzory adres pro autentizační funkce poskytované rámcem Django, které lze dále upravit pomocí šablon. Poslední a největší odkazovaný modul se nachází v hlavní složce aplikace. V něm se nachází mapování pomocí regulárních výrazů pro všechny vlastní definované pohledy. Ty jsou dále rozdělené do dvou oddělených modulů, prvním je `views.py`, ve kterém se nachází standardní pohledové funkce pro zobrazení celé webové stránky. Druhým modulem s pohledovými funkcemi je `ajax.py`, v němž se nachází funkce pro asynchronní požadavky, které vrací jen konkrétní údaje, nebo části stránek, jako je například obsah HTML tabulky nebo chybová zpráva, viz ukázka kódu 5.4.

Výpis 5.3: Mapování adres k pohledovým funkcím

```
1  from django.contrib import admin
2  from django.urls import path
3  from django.conf.urls import include
4
5  urlpatterns = [
6      path('admin/', admin.site.urls),
7      path('oauth/', include('social_django.urls', namespace='social')),
8      path('accounts/', include('accounts.urls')),
9      path('', include('AF_configurator.urls')),
10 ]
```

Funkce, které bylo vhodné generalizovat, či jsou potřebné v obou modulech s definicemi pohledů, jsou dále rozdělené v několika dalších modulech, jedná se hlavně o soubory `context.py`, `emails.py`, `filters.py`, `forms.py` a v sekci 5.1.1 již představený modul `models.py`, který obsahuje definice tříd databázových modelů.

V modulu `context.py` se nachází pro tento projekt elementární pomocná funkce, které slouží k získání kontextu pro renderování pohledů, které jsou zapotřebí pro základní šablonu, na které staví všechny ostatní. Mimo to je zde ještě jedna funkce, která slouží pro získání mnohem komplexnějšího kontextu pro administrační účely, jako je např. správa objednávek.

Modul `emails.py` zabaluje a odděluje práci se souborem pro přihlašování odběru informací emailem o změnách v instancích tříd databázových modelů od zbytku projektu. Mimo to je zde také funkce určená pro zaslání aktivačního emailu.

Jak název napovídá, `filters.py` obsahuje různé funkce pro filtrování. Oddělení těchto funkcí bylo inspirované primárně asynchronními požadavky. Chtěl jsem zde docílit toho, aby filtrování probíhalo, pokud možno, pomocí asynchronních požadavků a nebylo tak nutné znovu načítat stránku. Zároveň bylo také žádoucí, aby byly argumenty posílány pomocí metody typu GET, v takovém případě se totiž vyskytují argumenty přímo v URL adrese. Lze tak například poslat kolegovi odkaz s již vyplněnými argumenty pro filtrování, aby se mohl podívat na něco konkrétního.

Neméně důležitý je modul `forms.py`, ten obsahuje definice všech používaných formulářů, pro které poskytuje Django podporu již v základu. Ty jsou často ve dvou variantách – běžný a administrační. Volba mezi nimi se v takovém případě rozlišuje na základě oprávnění uživatele, která již byla probrána v sekci 5.1.2.

5.1.5 Internacionalizace a lokalizace

Při implementaci bylo nutné řešit internacionalizaci projektu. K tomu Django poskytuje řadu funkcí, jednu z nich lze vidět v kódu 5.4, jedná se o funkci `ugettext`, což je rozšířená funkce `gettext` o schopnost práce s unicode znaky. Způsobem, který je vidět v kódu, tak musí být obalené všechny řetězce, které jsou určeny k překladu. Tato funkce nicméně překládá řetězce pouze při jejich inicializaci, nefunguje tak správně například u textů v definici tříd. V takovém případě je nutno používat odvozenou funkci `ugettext_lazy`, která překládá řetězec při jeho každém použití.

Výpis 5.4: Asynchronní požadavek pro validaci uživatelského jména

```
1  from django.contrib.auth.models import User
2  from django.http import JsonResponse
3  from django.utils.translation import ugettext as _
4
5
6  def validate_username(request):
7      username = request.GET.get('username', None)
8      data = dict()
9      if User.objects.filter(username=username).exists():
10         data['error_message'] = _('User with this username already exists')
11
12     return JsonResponse(data)
```

Pro překlad textů v šablonách je zapotřebí nejdříve načíst balíček pro internacionalizaci, toho je docíleno příkazem `{% load i18n %}`, který musí být umístěn na začátku každého souboru šablon, ve kterém se nachází texty k překladu. Konkrétní řetězce pro překlad je poté nutné obalit v konstrukci `{% trans "Text k překladu"%}`, přičemž text k překladu nemusí být statický, může mít i podobu proměnné.

Pro lokalizaci je zapotřebí vygenerovat a následně zkompilovat soubory s překlady. Generování souborů proběhne příkazem `django-admin makemessages`. Django systematicky prozkoumá všechny soubory projektu a vytvoří soubory `django.po` pro každý definovaný jazyk. Ty obsahují dvojice řetězců pro každý nalezený text k překladu s využitím již dříve definovaných překladů s komentáři, značící jejich výskyt v kódu. Další dvojice lze přidat i ručně, čehož je nutno využít hlavně při lokalizaci textů, které se například vyskytují pouze

v databázi a v rámci projektu se tak vyskytnou jedině v proměnných. Zároveň jsou automaticky zakomentovány vyřazené překlady textů, které již nebyly nikde nalezeny. To vyvolává problém pro překlady dynamických řetězců v proměnných, které jsou vždy zakomentovány, jelikož v samotné aplikaci neexistují, je tak nutné je na konci souborů vždy po jejich novém vygenerování odkomentovat. Po sepsání všech překladů se `django.po` soubory zkompilují příkazem `django-admin compilemessages` do souborů `django.mo`. Tímto okamžikem se definované překlady začnou používat.

5.1.6 Záznamy změn v databázi a jejich odběr

Definici třídy pro zaznamenávání změn lze vidět v ukázce kódu 5.1. Jak bylo zmíněno v podkapitole 3.3, ukládány jsou údaje, pomocí kterých se lze odkázat na konkrétní instanci libovolné jiné třídy databázového modelu, změněná data, datum a čas změny a kdo změnu inicioval.

Jak bylo řečeno v sekci 5.1.1, aktuální uživatel nelze ukládat snadno automatizovaně, je tedy nutné jej do instance databázové třídy zapsat explicitně. Pro tyto potřeby jsem zvolil variantu, kdy se metodě třídy předávají data, v nichž figuruje i aktuálně přihlášený uživatel. V této metodě proběhne vytvoření záznamu s explicitním přiřazením uživatele a ostatních údajů a odeslání emailových zpráv s informací o vytvoření záznamu změny na adresy, které jsou registrované k odebírání informací o změnách instancí zaznamenávaného databázového modelu.

Pomocí kombinace dat `app_label` a `table_name` lze jednoznačně určit a získat třídu databázového modelu. Hodnotu `field_pk` pak lze využít k získání konkrétní instance dané třídy z databázového modelu. Tyto údaje jsou stejně tak používané ve funkcích modulu `emails.py` pro zápis a čtení seznamu odběrů do souboru ve formátu JSON. Při volání těchto funkcí se soubor automaticky doplní o pole pro možné odběry upozornění o změnách instancí tříd modelu, jejichž změny byly zaznamenány již v minulosti. Dále je v souboru specifikováno přihlášení odběru všech změn, čímž si lze zajistit obdržení upozornění na změny i v dříve nezaznamenávaných databázových třídách.

Konkrétní změněné údaje instancí hlídaných tříd databázového modelu jsou uloženy ve formátu JSON v poli data. Je zde ukládána sada klíčů a dvojic hodnot pro každý klíč, kde klíč je název měněného pole v měněné instanci databázového modelu, dvojice hodnot pak zaznamenávají hodnotu, nebo její vhodnou reprezentaci před změnou a po změně. Pokud se v měněném poli jedná o cizí klíč, tak by například reprezentace pouhým číslem nebyla pro uživatele, který si bude záznam změn prohlížet, příliš přehledná, a tedy ani vhodná. V takovém případě se ukládá např. u subjektů HTML odkaz směřující na webovou stránku pro správu subjektu s textem názvu onoho subjektu.

5.1.7 Práce s vektorovou mapou zóny

Při přidání nové zóny, které jsou reprezentovány vektorovou mapou ve formátu SVG (Scalable Vector Graphics), dochází k automatickým úpravám souboru mapy, které mají zaručit co největší kompatibilitu. Podporovány tak jsou různé polygony a útvary, které jsou reprezentovány buď pomocí bodů (atribut `points`), nebo pomocí křivek (atribut `d`). Body křivek jsou definovány parametry typu `M` (`Move`) a `L` (`Line`), tyto parametry mohou být v případě potřeby také automatizovaně sloučeny.

Při nahrání je každému objektu s grafickou reprezentací (tedy takovému, který má jeden z výše jmenovaných parametrů) přiřazen v rámci souboru jednoznačný identifikátor, ten je později využíván, pro odkazování při definici rezervovatelných míst. Místům definovaným

pro možnost rezervace je přidělen parametr `key`, který je uživatelsky definovaný a slouží jako ID v rámci mapy, v níž tedy musí být taktéž unikátní. Jedná se jednak o příznak definované oblasti, ale také o reprezentační hodnotu v textové podobě.

5.2 Klientská část (Front-end)

V této podkapitole bude řešena klientská část (angl. front-end) aplikace. Jedná se o část s uživatelským rozhraním aplikace, pomocí kterého uživatelé komunikují se serverem a vyměňují si s ním data.

V první části se bude tato podkapitola věnovat Django jazyku šablon, který se pokusí s využitím ukázek popsat, další sekce pak bude věnována některým z důležitějších pohledů grafického uživatelského rozhraní webové aplikace.

5.2.1 Django jazyk šablon

Django jazyk šablon (angl. Django template language) je jazyk poskytovaný aplikačním rámcem Django. Tento jazyk slouží pro dynamické generování textově založených formátů (jako např. HTML či XML) za pomoci tzv. šablon (angl. template), kterých lze využívat i větší množství při jednom vykreslování Django pohledu.

Pomoci údajů HTTP žádosti a dalších dat proměnných předaných z Django pohledu šablonám tak lze rozhodovat například které části stránky se mají zobrazit, nebo třeba generovat unikátní obsah na základě přihlášeného uživatele.

Výpis 5.5: Šablona se společným základem stránek

```
{% load staticfiles %}
{% load i18n %}

<html lang="{{ lang }}">
<head>
    <link rel="stylesheet" href="{% static 'css/style.css' %}">
    <title>{% block title %}Animefest{% endblock %}</title>
</head>
<body>
    <div class="content">
        {% block messages %}
            {% for message in messages %}
                <p>{% trans message %}</p>
            {% endfor %}
        {% endblock %}
        {% block content %}{% endblock %}
    </div>
    <footer class="footer">
        <a href="{% url 'tos' %}">Terms of Service</a>
        <a href="{% url 'privacy_policy' %}">Privacy policy</a>
    </footer>
</body>
</html>
```

V ukázce kódu 5.5 lze vidět, jak by mohla například vypadat velmi jednoduchá šablona pro základ stránky. Na začátku takového souboru se nachází příkazy `{% load %}`, kterými se načítají pro šablonu potřebná rozšíření. `staticfiles` zde slouží pro automatické generování adres statických souborů, to lze vidět při odkazování se na soubor `css/styles.css`. Načtení `i18` zase povoluje používání bloků typu `{% trans %}` pro překlady, přičemž tímto způsobem lze překládat proměnné, jak je vidět přímo v ukázce, nebo statické řetězce. Dále zde lze vidět generování obsahu pomocí cyklu `{% for message in messages %}`, nebo generování URL adres pomocí reverzního mapování názvů Django pohledů na adresy. Asi nejdůležitější pro základ stránky jsou ale bloky typu `{% block content %}`, pomocí nich můžeme rozšiřovat šablonu v místech těchto bloků z jiných šablon. V případě, že tento typ bloku není prázdný, znamená to, že se jedná o implicitní obsah, který lze v jiné šabloně předefinovat.

Ukázka 5.6 demonstruje, jak lze využít základní šablony pro vytvoření komplexnější stránky. Na začátku souboru je tentokrát údaj o šabloně, která se má použít jako základ, přičemž většina bloků základní šablony zůstala netknutá, rozšiřován je pouze blok `content`. V něm je předvedeno využití dalších šablon, které lze snadno vzít a použít pro komplexní vykreslení. Tímto způsobem lze zabránit přílišné duplikaci kódu nebo rozdělit kód do více souborů a ty tak udržet přehlednější.

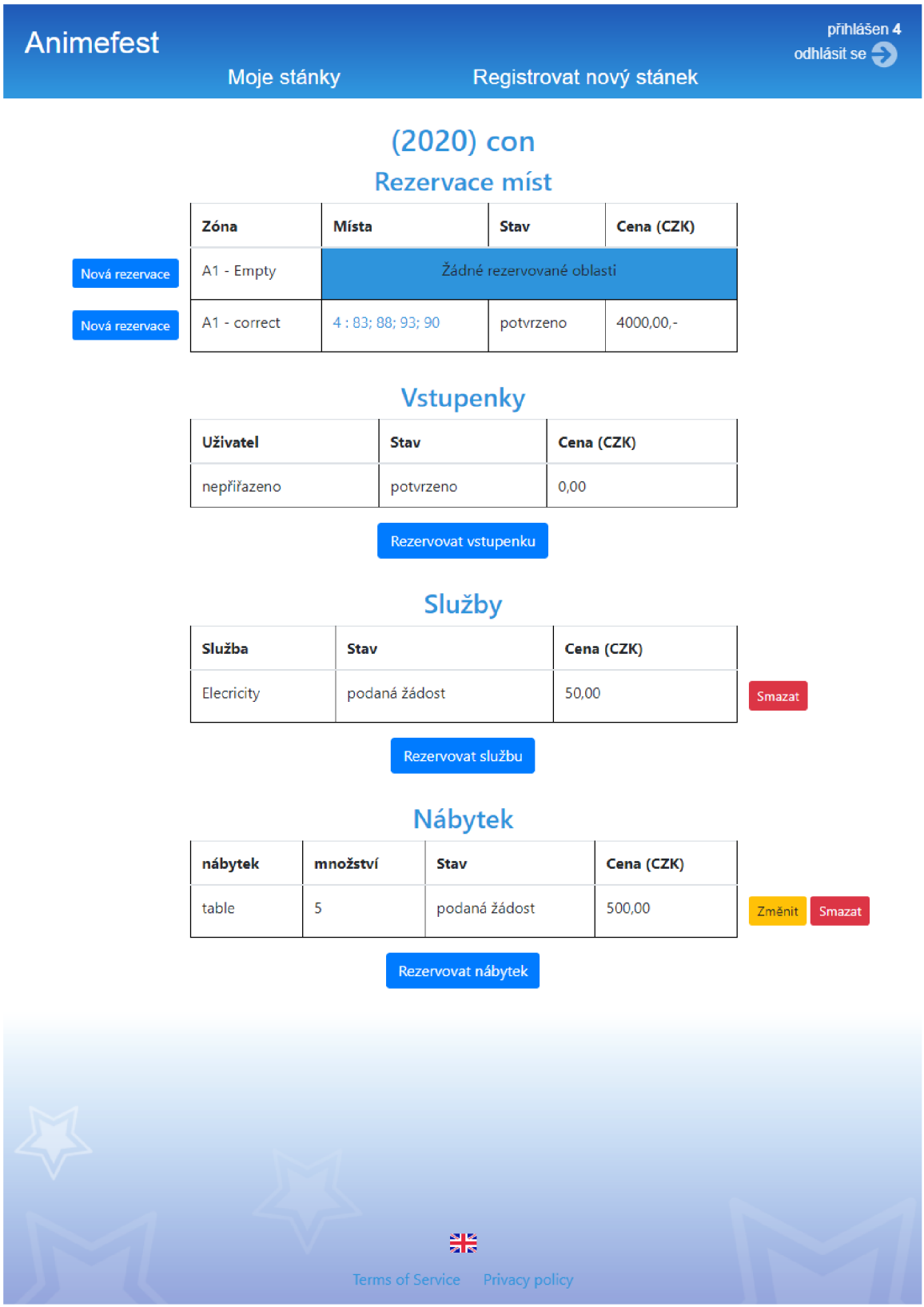
Výpis 5.6: Rozšiřující šablona

```
{% extends "base.html" %}
{% load staticfiles %}

{% block content %}
    {% include "manage/filters.html" %}

    <table id="table_list">
        {% include "manage/header.html" %}
        {% include "manage/list.html" %}
    </table>
{% endblock %}
```

5.2.2 Grafické uživatelské rozhraní aplikace

V grafickém uživatelském rozhraní (angl. Graphic User Interface) této webové aplikace byl kladen důraz primárně na možnost získat efektivně větší objem informací a udržení jednoduchosti ve struktuře pro co nejsnazší zorientování.

Z výše zmíněných důvodů byly v průběhu vývoje například sloučeny zpočátku samostatně vyvíjené Django pohledy pro správu různých typů objednávek pro konkrétní stánek. Výsledek tohoto sloučení lze vidět na obrázku 5.1, kde se již nachází přehled všech dosavadních objednávek daného stánku společně. Pro pohodlí uživatelů byly taky zpočátku samostatné stránky s formuláři pro rezervace vpraveny do tohoto pohledu za využití asynchronních požadavků pracujících s modálním oknem. Za předpokladu, že uživatel nemá extra oprávnění, tlačítka pro úpravu rezervací a tvorbu nových rezervací nebudou nadále zobrazována po uzavření stánku pro finalizaci a vyúčtování.



Obrázek 5.1: Pohled pro správu objednávek stánku.

Vektorová mapa a procházení zón

Na obrázku 5.2 lze vidět pohled pro procházení zón a jejich podsekcí, pro což je využívána vektorová mapa ve formátu `SVG`. V tomto informativním pohledu bylo nutné dbát mimo jiné také na dostatečně kontrastní barevné rozlišení různých stavů oblastí definovaných pro možnost rezervace, toho je docíleno pomocí přiřazení tříd na základě informací o oblastech pomocí aplikačního rámce jQuery.

Tento pohled by měl být využíván primárně během konání akce pro možnost rychlého vyhledání informací o stánku na základě jeho umístění. V tomto případě jsou načteny všechny možné potřebné údaje již při načtení stránky a uloženy v proměnných v jazyce JavaScript, jelikož posílat asynchronní žádosti o data by během konání akce vedlo na nežádanou prodlevu a mohly by nastat problémy při horší stabilitě internetového připojení. Pro zobrazení dat tímto způsobem stačí teoreticky jen načtená stránka a není zapotřebí být nadále připojen k internetu.

Detail informací o vybraném místu se v původní implementaci zobrazoval nad samotnou mapou, dle uživatelské zpětné vazby však bylo příliš rušivé posouvat stránku nahoru a zpět kvůli těmto informacím. Aktuálně se proto informace zobrazují v modálním oknu, které se otevře výběrem konkrétního místa.

Do budoucna by se z tohoto pohledu mohla vyvinout a oddělit také interaktivní mapa vhodná pro návštěvníky akce. Návštěvníci by mohli totiž naopak chtít vyhledávat hlavně lokaci konkrétních stánků, nebo si rychle vyhledat blízké stánky s občerstvením.

Výpis záznamů změn

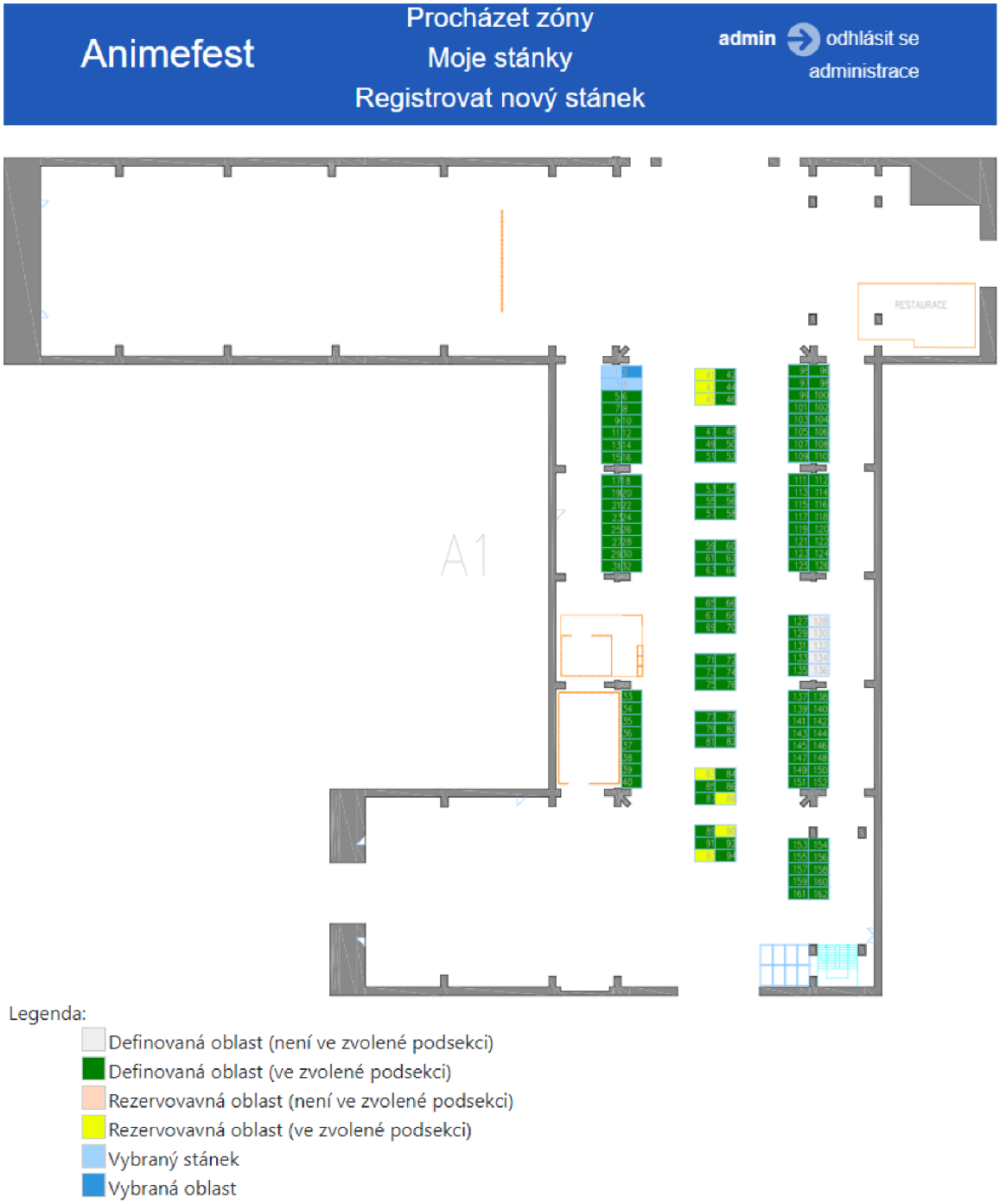
Na obrázku 5.3 je vidět pohled s detailem jednoho záznamu databázových změn. Jedná se tedy o pohled pro zobrazení záznamů změn různých databázových tříd, které byly zhotoveny pomocí vlastní třídy, jejíž návrh byl řešen v podkapitole 3.3 a jejíž implementaci se věnuje sekce 5.1.6.

V pohledu bylo nutné hlavně zvýraznit změněná data, aby bylo možné snadno rozpoznat, na které údaje zaměřit svou pozornost. Jelikož však jde hlavně o snadné dohledávání informací, tak tento přístup samotný nebyl dostatečný, proto lze v jednom snímku obrazovky vidět více různých tabulek.

Druhá tabulka v pořadí obsahuje všechny záznamy související s konkrétním prohlíženým objektem, lze tak snadno najít kterýkoliv v historii zaznamenaný stav.

Následující tabulka představuje výpis konkrétních změn tak, jak byly za sebou zaznamenány, což se může hodit pro nalezení konkrétní změny, kdy nastala nějaká uživatelská nebo systémová chyba.

Jelikož se jedná o záznam stánku, v poslední tabulce lze nalézt ještě výpis záznamů jiných stánků, které spadají pod stejného majitele.



Obrázek 5.2: Pohled pro procházení zón a jejich podsekcí.

Value	Before	After	Now
id	36	36	36
owner_id	38	38	38
section_id	Craft Alley	Craft Alley	Craft Alley
year	2020	2020	2020
vat	None	None	None
vat_payer	False	False	False
ico	None	None	None
title	con	con	con
address	None	None	None
email	4@seznam.cz	4@seznam.cz	4@seznam.cz
phone	4	4	4
info			
link	None	None	None
logo	None	None	None
photo	None	None	None
country	None	None	None
price	0.00	0.00	0.00
paid	0.00	0.00	0.00
notes			
notes_int	None	None	None
lang	cs	cs	cs
state_id	requested	confirmed	confirmed
date	2020-07-03 20:10 p.m.	2020-07-03 20:10 p.m.	2020-07-03 20:10 p.m.

Logs of this object

User	Field	Data	Date
admin	(2020) con	state_id: requested -> confirmed	2020-07-20 15:03 p.m.
admin	(2020) con	confirmed: False -> True	2020-06-16 16:02 p.m.

Changelog

User	Value	Before	After	Date
admin	state_id	requested	confirmed	2020-07-20 15:03 p.m.
admin	confirmed	False	True	2020-06-16 16:02 p.m.

Logs of other stalls of this user

User	Field	Data	Date
Veve	(2020) Uncon	state_id: requested -> confirmed	2020-07-29 17:42 p.m.
admin	(2020) closed	state_id: requested -> closed	2020-07-20 16:29 p.m.

Obrázek 5.3: Pohled pro zobrazení detailu záznamů změn instance databázové třídy.

Kapitola 6

Testování aplikace

V této kapitole bude popsána nezbytná část ve vývoji informačního systému, ale i kteréhokoliv jiného typu programu, tím je testování. Hlavním účelem testování je zkoumání aplikace za účelem odhalení chyb a zvýšení kvality software jejich nápravou, toto testování probíhá po celou dobu vývoje, v sekci 6.1 lze nalézt jeho shrnutí.

Dalším cílem testování je vyzkoušení systému jeho potenciálními budoucími uživateli, získání jejich reakcí a vylepšení software dle této zpětné vazby pro maximalizaci komfortu uživatelů. Tato část testování bude probrána v sekci 6.2.

6.1 Testování programátorem při vývoji

Během vývoje procházela aplikace pravidelným testováním, a to především při vývoji každé nové funkcionality a úpravách již stávajících funkcionalit. V některých případech však musel být upraven dřívější návrh. K tomu obvykle docházelo buď z důvodu nepochopení se při detailní specifikaci požadavků, nebo kvůli naražení na nějaký pouze obtížně řešitelný problém, se kterým by byly větší potíže, než s pozměněním samotného návrhu.

Pro potřeby testování byly mimo jiné vyvinuty i některé samostatné funkce a pohledy. Mezi takové lze zařadit například pohled, který odesílá email na adresu uživatele. Tento pohled byl využíván hlavně při nastavování potřebných proměnných pro umožnění odeslání samotného emailu. Tento přístup v cíli významně usnadnil a urychlil nejen testování, zda je nastavení správné a emaily jsou odesílány, ale i jestli jsou emaily správně formátovány.

Tato část testování probíhala primárně pod administrátorským účtem, jelikož testování pod každým typem uživatelské role a jejich kombinací by zabralo příliš mnoho času. Komplexnější test pod všemi definovanými uživatelskými rolemi proběhl proto až v pozdější fázi vývoje, synchronně s ním byl sepisován základní uživatelský manuál, který měl být poskytnut pro následující uživatelské testování. Větší část testů probíhala lokálně, nešlo to však vždy. Například aplikace třetích stran, jako je třeba Twitter, požadovaly pro umožnění autentizace uživatelů nejdříve nastavit doménu, na které bude autentizace využita. Jelikož však blokují adresy typu localhost, nelze tuto funkci testovat lokálně bez veřejné IP adresy. Z toho důvodu probíhala tato část testování rovnou na serveru.

6.2 Uživatelské testování

Ve finální fázi vývoje proběhlo uživatelské testování, kterého se zúčastnili někteří z potenciálních budoucích uživatelů vyvíjeného informačního systému. Dále byli k testování přizváni

také vybraní jednotlivci, hlavně z řad studentů informačních technologií. Toto testování si kladlo za cíl primárně otestovat přehlednost grafického uživatelského rozhraní. Dalším cílem v pořadí bylo odhalení dříve přehlédnutých chyb, které se nemusely projevit s ohledem na specifické chování programátora, který se k systému chová dle vlastního návrhu, nikoliv dle toho, k čemu systém vybízí se systémem neseznámeného člověka.

Před začátkem samotného testování byl pro potenciální budoucí uživatele zhotoven krátký uživatelský manuál, který by měl sloužit pro navedení k důležitým funkcím systému. Tento manuál je přiložen na paměťovém médiu. Testování se následně účastnilo kolem dvaceti respondentů, kterým byl rozeslán také dotazník pro zjištění spokojenosti a případných výtek. Dotazník vyplnilo celkem 16 uživatelů, dále budou zmíněny některé ze zjištěných výsledků a případných implementovaných změn, kompletní dotazník se získanými daty je přiložený na paměťovém médiu.

S většinou uživatelů testujících systém bylo komunikováno přímo během samotného testování, což umožnilo rychlejší opravu chyb a zavádění objektivně vhodnějších navržených řešení. Díky tomuto rychlému nasazování změn se nemuseli všichni uživatelé potýkat se stejnými problémy a vyrušujícími prvky, což pomohlo získat větší rozptyl odpovědí. Dotazníky však vůči odpovědím při osobním přístupu zaznamenávají spíše menší část zpětné vazby. Některé odpovědi v dotaznících byly ještě navíc konzultovány s jejich autory.

V dotazníku bylo zjišťováno například, jestli narazili uživatelé na nějaký problém s responzivitou, opakovaně byl uváděn problém překrývajících se textů v hlavním menu na telefonech, jelikož se jednalo o častý problém, bylo vhodné jej řešit. Po úpravách byli někteří z uživatelů, kteří problém uvedli znovu dotázáni a problém se zdá vyřešen.

Dále byl zkoumán názor uživatelů na aktuální rozdělení správy různých entit (vstupenek, nábytku, služeb...), přičemž se takřka všichni shodují, že rozdělení je vyhovující a nechtěli by slučovat správu více entit do jednoho pohledu.

V zadání této práce byla také lokalizace do češtiny a angličtiny, proto bylo nutné vložit do uživatelského rozhraní přepínač jazyka. Rozhodl jsem se pro jednoduchou vlaječku, která se nachází ve spodní části stránky. Cílem tohoto umístění je, aby nepůsobila rušivě. Dotazník poté zkoumal, jestli není vlajka až příliš nenápadná. Čtvrtina uživatelů uvedla, že si jí vůbec nevšimla, další čtvrtina, že si jí všimli až po delší době. Z osobních rozhovorů však vyplynulo, že většina uživatelů se domnívá, že by si jí všimli, kdyby vážně hledali volbu pro změnu jazyků.

V původní implementaci byly znázorňovány booleovské hodnoty pomocí barevně neutrálního HTML prvku pro zaškrtávací políčko (angl. checkbox), které bylo deaktivované. Jelikož se však nabízela možnost využití barevných unicode znaků, tak byli tázáni samotní uživatelé, kterou variantu preferují, přičemž panuje stoprocentní shoda v preferenci barevných znaků. Tento detail byl tedy upraven.

Při následné kontrole záznamů bylo zjištěno, že někteří aktivnější uživatelé zkoušeli testovat také zabezpečení informačního systému. Zkoušeli tak například podvrhnout databázový příkaz, nebo kód v jazyce JavaScript. Databázové příkazy si hlídá samotný aplikační rámec Django, s tím tedy nebyl problém. Kód v jazyce JavaScript však mohl být za určitých okolností automaticky spuštěn, pokud byl součásti záznamů změn. Tato bezpečnostní trhlina byla následně napravena.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo vytvořit informační systém, který by usnadnil plánování rozvržení výstavních ploch během festivalu Animefest a pomohl do budoucna přehledněji udržet minulé zkušenosti s prodejci.

Na počátku práce proběhlo několik schůzek s organizátory festivalu. Během těchto schůzek jsem byl obeznámen s aktuálně fungujícími procesy při registraci partnerů a s požadavky na informační systém, který by měl tyto procesy usnadnit.

Na základě analyzovaných požadavků bylo možné navrhnout informační systém. Základ tohoto návrhu stavěl na entitně-vztahovém diagramu, který znázorňuje abstrakci poměrně komplexní databáze s mnoha entitami a vazbami mezi těmito entitami. Dále byla provedena analýza dostupných technologií, na základě níž bylo rozhodnuto pro implementaci pomocí programovacího jazyku Python 3 a aplikačního rámce Django.

Návrh byl implementován jako webová aplikace a během vývoje průběžně testován autorem. V závěru práce bylo také provedeno uživatelské testování se základnou kolem dvaceti uživatelů. Na jeho základě byly opraveny dříve neodhalené chyby a implementovány některé návrhy na zpříjemnění uživatelského rozhraní. Vzhledem ke zrušení akce vlivem koronaviru však systém nemohl být nasazen do ostrého provozu, a to ani v předběžné neúplné verzi.

Výsledná aplikace využívá vektorovou mapu, na jejímž základě definuje prostory, které následně umožňuje spravovat. Pro samotnou práci s mapou je imeplementováno více různých pohledů. Do budoucna by mohlo být dobré práci s touto mapou ještě více rozšířit a umožnit například jednoduché posouvání jednotlivých prvků mapy. Pro zákazníky by taky mohlo být zajímavé využití základu jednoho z již implementovaných pohledů a adaptovat jej na interaktivní mapu pro potřeby zákazníků.

Při rezervaci prostor by mohla do budoucna taky pomoci možnost vhodným způsobem nadefinovat, které oblasti lze rezervovat za jakých okolností, například za pomoci definování sousednosti míst. To by mohlo vést ke zvýšení pravděpodobnosti partnera na přijmutí jeho rezervace správcem prostor, a tedy dále optimalizovat průběh rezervací prostor.

Literatura

[1] *Django Vs Laravel: Which framework to choose?* [online]. [cit. 2019-07-16]. Dostupné z: https://hackr.io/blog/django-vs-laravel.

[2] Animefest. *Oficiální stránky festivalu Animefest.* [online]. [cit. 2019-01-02]. Dostupné z: http://www.animefest.cz.

[3] COMM/DG/TAXUD. *Ověřování DIČ pro účely DPH prostřednictvím systému VIES* [online]. [cit. 2019-07-23]. Dostupné z: https://ec.europa.eu/taxation_customs/vies/vieshome.do.

[4] Django Software Foundation. *Django 2.2 databases* [online]. [cit. 2019-07-09]. Dostupné z: https://docs.djangoproject.com/en/2.2/ref/databases/.

[5] Django Software Foundation. *Django 2.2 documentation* [online]. [cit. 2019-01-01]. Dostupné z: https://docs.djangoproject.com/en/2.2/.

[6] Django Software Foundation. *Django views* [online]. [cit. 2019-07-24]. Dostupné z: https://docs.djangoproject.com/en/2.2/topics/http/views/.

[7] Django Software Foundation. *Internationalization and localization* [online]. [cit. 2019-07-24]. Dostupné z: https://docs.djangoproject.com/en/2.2/topics/i18n/.

[8] Django Software Foundation. *Supported Versions* [online]. [cit. 2019-06-29]. Dostupné z: https://www.djangoproject.com/download/.

[9] GeeksforGeeks. *Difference between MVC and MVT design patterns* [online]. [cit. 2019-07-26]. Dostupné z: https://www.geeksforgeeks.org/difference-between-mvc-and-mvt-design-patterns/.

[10] JS Foundation - js.foundation. *JQuery documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://api.jquery.com/.

[11] JS Foundation - js.foundation. *JQuery UI* [online]. [cit. 2019-07-24]. Dostupné z: https://jqueryui.com/.

[12] Mark Otto, Jacob Thornton, and Bootstrap contributors. *Bootstrap documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://getbootstrap.com/docs/4.5/getting-started/introduction/.

[13] Mozilla Developer Network and individual contributors. *CSS documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://devdocs.io/css/.

[14] Mozilla Developer Network and individual contributors. *HTML documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://devdocs.io/html/.

[15] Mozilla Developer Network and individual contributors. *JavaScript documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://devdocs.io/javascript/.

[16] Python Software Foundation. *Python 3.7.5 documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://stackless.readthedocs.io/en/3.7-slp/.

[17] The PostgreSQL Global Development Group. *PostgreSQL documentation* [online]. [cit. 2019-07-01]. Dostupné z: https://www.postgresql.org/docs/.

[18] The Psycopg Team. *PostgreSQL driver for Python — Psycopg* [online]. [cit. 2019-07-09]. Dostupné z: https://www.psycopg.org/.

Příloha A

Postup instalace pro operační systém Ubuntu

Pro zajištění fungování aplikace je nejprve zapotřebí nainstalovat všechny závislosti, toho lze docílit následující sekvencí příkazů:

```
$ sudo apt-get install python3.7
$ sudo apt-get install python3-pip
$ pip3 install "django>=2.2,<3"
$ sudo apt-get install python3-psycopg2
$ pip3 install Pillow
$ pip3 install django-widget-tweaks
$ pip3 install social-auth-app-django
$ pip3 install django-vies
```

Dále je nutno doplnit platnou konfiguraci pro připojení k databázi, emailu a autorizačním službám v modulu settings.py ve složce Animefest. Potřebné proměnné jsou doplněny komentářem TODO.

Příloha B

Uživatelský manuál

Tento manuál má za účel ve zkratce popsat, jak se má se systémem pracovat a co je možné dle role uživatele.

B.1 Základní uživatel

V pravé horní části stránky lze vidět vlastní login a možnost pro odhlášení, po kliknutí na login se otevře profil uživatele, odkud lze měnit své údaje a přidat/odebrat možnosti přihlášení pomocí aplikací třetích stran (Twitter, Google).

Základní uživatel může spravovat stánky, k nimž byl přiřazen. Přičemž pokud nemá zvýšená práva, může si jen prohlížet data stánku, aby mohl mít připomínky pro majitele. Přiřazené stánky najde pod možností „Moje stánky“ (angl. „My stalls“) v hlavním menu. Tuto možnost neuvidí, pokud nemá přístup k údajům žádného stánku.

B.2 Zodpovědná osoba (angl. Responsible person)

Uživatel s touto rolí může vytvářet nové stánky v systému. Registrace probíhá vyplněním formuláře pod možností v hlavním menu „Registrovat nový stánek“ (angl. „Register new stall“). Při registraci musí zvolit tzv. sekci stánku, která omezuje datum registrací do specifikovaného data.

Nově registrovaný stánek se bude nacházet ve stavu „requested“, do jeho schválení nelze vytvářet žádné rezervace. Po otevření detailu stánku (možnost nalezitelná pod „Moje stánky“ v hlavním menu) lze na vrcholu stránku vidět dva odkazy, „Personál“ (angl. „Staff“) a „Rezervace“ (angl. „Reservations“).

V pohledu personálu je jednoduché pole pro přidání uživatelů, kam je zapotřebí zadat jejich uživatelské jméno nebo email. Přidaným uživatelům lze dále dát oprávnění správce stánku, čímž jim bude umožněno upravovat údaje stánku a po jeho schválení také provádět rezervace.

V pohledu rezervací lze přidávat nové rezervace, pokud je stánek ve stavu „schválený“ (angl. „confirmed“). Jedná se o rezervace míst, vstupenek, služeb a nábytku. Nové rezervace se nachází ve stavu „requested“, dokud jsou v tomto stavu, mohou je správci daného stánku zrušit či upravit.

V případě, že má daná zodpovědná osoba nějaké registrované stránky z minulých let, může si otevřít jejich detail a využít jeho data pro novou rezervaci pro aktuální ročník.

B.3 Správce sekcí (angl. Section manager)

Účelem správce sekcí je potvrzování registrací stánků. V pravém horním rohu stránky může vidět administrační panel (volby se ukážou po najetí na text „administrace“).

Správa stánků (angl. Manage stalls)

Správa stánků je primární účel této role. Stánky lze filtrovat pomocí jednoduchého formuláře v horní části stránky.

Po výběru stánků je zde lze schvalovat (volba „Potvrdit“), zamítnout, či pro již potvrzené stánky uzavřít objednávky za účelem finalizace vyúčtování.

Z tabulky stánků lze otevřít detail majitele stánku (sloupec „Majitel“ – angl. „Owner“), detail registrace stánku (sloupec „Stánek“ – angl. „Stall“), či si zobrazit krátký výpis ukazující celkovou cenu objednaných prostor, nábytku, vstupenek a služeb spolu s udělenými pokutami. Ukazují se přitom dvě různé sumy – schválené objednávky a schválené spolu s takovými, které teprve čekají na vyřízení. Pokud se neshodují je v tabulce pole s cenou podbarvené jako upozornění kvůli vyúčtování.

B.4 Správce sekcí a správce prostor

Popis pohledů, které mají tito uživatelé společné pod administračním panelem v pravém horním rohu stránky.

Správa uživatelů (angl. Manage users)

Pod volbou správy uživatelů lze nalézt seznam všech uživatelů aplikace, jejich základní informace a možnost. Uživatele lze filtrovat formulářem v horní části stránky, například lze zadat, zda se mají zobrazit pouze ověření uživatelé, neověření, či na tomto jevu nezáleží. Po výběru uživatelů lze zvolit akci (pod výpisem uživatelů) a tu provést, může se jednat o deaktivaci/aktivaci, zrušení ověření/ověření, či přidělení role zodpovědné osoby, čímž umožní dané osobě registrovat stánky.

Pokuty (angl. Penalties)

Tento pohled má pouze pro správce sekcí a prostor pouze informativní hodnotu, v horní části stránky se opět vyskytují možnosti filtrování. Dále lze zkontrolovat kdo jakému stánku udělil pokutu (přidělování pokut je v kompetenci pouze administrátora), případně pokud byl označen nějaký viník pro pokutu. Rovněž lze otevřít detaily uživatelů a stánků.

Logy (angl. Logs)

Další pohled s informativní hodnotou, lze kontrolovat prováděné změny v systému na základě řady filtrů, lze zobrazit detail uživatelů či detail konkrétního logu (odkaz ve sloupce „Pole“ – angl. „Field“), poté lze vidět související záznamy, které by mohly uživatele zajímat.

Pod tabulkou lze nalézt odkaz „Přihlášení k odběru“ (angl. „Subscription“) odkud se lze přihlásit k odběru informací o zaznamenaných změnách na email.

B.5 Správce prostor (angl. Space manager)

Účelem správce prostor je spravovat prostory a objednávky jednotlivých stánků. V pravém horním rohu stránky může vidět administrační panel (volby se ukážou po najetí na text „administrace“). Jednotlivé možnosti správy budou probrány v této sekci, možnosti správy uživatelů, a prohlížení pokut a logů lze nalézt v předchozí sekci (Správce sekcí a správce prostor).

Správa oblastí (angl. Manage areas)

Tento pohled slouží pro schvalování a zamítání rezervací oblastí pro jednotlivé stánky. Lze vidět základní přehled objednávek prostor pro všechny schválené stánky.

Při schválení rezervace prostor se za každou oblast v ní vygeneruje již schválená vstupenka zdarma. Zamítnutí uvolní místa, která byla rezervována pro jiné rezervace. Nelze tedy schválit již zamítnutou rezervaci pro možný konflikt s novější rezervací. Zamítnutí dříve schválené rezervace uvolní místa, nicméně nezruší vstupenky, to je v takovém případě nutno udělat ručně!

Po kliknutí na název stánku v tabulce se lze dostat na stránku s detailem objednávek daného stánku, odkud je lze upravovat (včetně ceny) či přidávat.

Správa vstupenek, nábytku, služeb (angl. Manage tickets, furniture, services)

Tyto pohledy slouží pro schvalování a zamítání rezervací daného typu pro jednotlivé stánky. Lze zde vidět základní přehled objednávek pro všechny schválené stánky.

Po rozkliknutí odkazu pod jménem stánku v tabulce se lze dostat na stránku s detailem objednávek daného stánku, odkud je lze upravovat (včetně ceny) či přidávat.

Správa zón (angl. Manage zones)

Tento pohled slouží k přidávání prostor a ke správě možností jejich objednání. V dolní části stránky se nachází tlačítko „Vytvořit novou zónu“ (angl. „Create new zone“), které přesměrovává na jednoduchý formulář, kam je nutno vyplnit název zóny (např. „Pavilon A1 – přízemí“) a přidat vektorovou mapu, po čemž se zobrazí pohled pro definování první podsekce, který je ale možno opustit a vrátit se později.

Jakmile je definovaná alespoň 1 zóna, lze vidět nalevo od tabulky s přehledem zón a jejich podsekcí tlačítka pro odstranění zóny (Vysoce nedoporučeno v provozu!) a tlačítko pro definici nové podsekce. Pokud již jsou nějaké podsekce definovány, vpravo od tabulky je možnost je tlačítko pro jejich smazání. (Smazání podsekce nemá vliv na aktuální objednávky ani definice míst a jejich cen, pouze dojde k omezení možností rezervací míst, které zde byly obsaženy.) Kliknutím na název podsekce dojde k přesměrování na její úpravu. Pomocí formuláře pod tabulkou lze upravit, které sekce stánků si mohou rezervovat místa obsažená v podsekci.

Při definici podsekce je nutno nejdříve kliknout na prvek mapy, na jehož základě se definují vybratelné objekty. Vybírat lze buď klikáním, nebo výběrem oblasti při držení levého tlačítka myši. U tohoto výběru je dříve nedefinovaným místům přiřazen identifikační klíč, který musí být unikátní v rámci mapy. V horní části stránky se nachází nastavení pro definici míst. Lze nastavit od jakého čísla bude probíhat označování a případně číslu přidat prefix, dále pak taky zda má docházet při výběru více míst najednou k číslováno po řádcích či sloupcích a zda má dojít k novému přiřazení ceny a identifikačního klíče u již dříve definovaných míst. Přehled vybraných míst lze vidět pod mapou, tlačítko pro odeslání formuláře (definování podsekce) se nachází ve spodní části stránky.

Příloha C

Obsah přiloženého média

- README.txt - Postup instalace
- manual.pdf - Uživatelská příručka
- dotaznik.zip - Exportovaný dotazník včetně odpovědí uživatelů
- test_map.svg - Testovací vektorová mapa
- text.pdf - Text práce
- text.zip - Zdrojové kódy pro překlad této práce
- src - složka se zdrojovými kódy aplikace