STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

Obor: 18. Informatika

Webový portál BurzaŠkol.Online

Vít Falta

Pardubice 2020

STŘEDOŠKOLSKÁ ODBORNÁ ČINNOST

WEBOVÝ PORTÁL BURZAŠKOL.ONLINE

BURZAŠKOL.ONLINE WEB PORTAL

AUTOR	Vít Falta
ŠKOLA	DELTA - Střední škola informatiky a ekonomie, s.r.o.
KRAJ	Pardubický
ŠKOLITEL	RNDr. Jana Koupila, Ph.D.
OBOR	18. Informatika

Pardubice 2020

Prohlášení

Prohlašuji, že svou práci na téma *Webový portál BurzaŠkol.Online* jsem vypracoval/a samostatně pod vedením RNDr. Jana Koupila, Ph.D. a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Dále prohlašuji, že tištěná i elektronická verze práce SOČ jsou shodné a 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 změně některých zákonů (autorský zákon) v platném znění.

V Pardubichích dne: _____ _____

Vít Falta

Poděkování

Děkuji svému školiteli RNDr. Janu Koupilovi, Ph.D. za obětavou pomoc, podnětné připomínky a nekonečnou trpělivost, kterou mi během práce poskytoval.

Anotace

Cílem této práce je vytvořit online platformu, která umožní nahrazení klasických burz škol v České Republice, pomocí moderních webových technologií v době pandemie Covidu-19. Tato platforma zprostředkuje kontakt mezi středními školami a žáky 9. ročníků základních škol v době, kdy jsou klasické burzy škol z hygienických důvodů nemožné.

Klíčová slova

burza škol; webová stránka; www; online; online platforma

Annotation

The goal of this work is to create an online platform that will allow the replacement of traditional school exchanges in the Czech Republic using modern web technologies during the Covid-19 pandemic. This platform mediates contact between secondary schools and pupils in the 9th grade of primary schools, at a time when traditional school exchanges are impossible due to hygienic restritions.

Keywords

school exchange; web page; www; online; online platform

Obsah

Úvod

Každý podzim se v České Republice konají tzv. Výstavy středních škol, nebo také Burzy, či Veletrhy vzdělávání (názvy se liší kraj od kraje). Hromadné akce na kterých se střední školy snaží zaujmout co nejvíce žáků 9. tříd základních škol – deváťáků a vylíčit svou školu v nejlepším světle.

Deváťáci mají možnost si v jeden čas a na jednom místě prohlédnout nabídky jednotlivých škol. Pokud je nějaká škola zaujme mají možnost získat podrobnější informace od zástupců dané školy.

Vzhledem k pandemické situaci roku 2020 spojené s nemocí COVID-19, ale nebylo možné tyto Burzy uspořádat. Systém *BurzaŠkol.Online*umožnil přesunutí celého náboru deváťáků do virtuálního prostředí internetu.

Jak to funguje?

*BurzaŠkol.Online*je internetový portál zprostředkující kontakt mezi deváťáky a středními školami za pomoci on-line video konferencí (nejčastěji MS Teams[1], Google Meet[2], Zoom[3], ...). Běžnému návštěvníkovi portál nabízí seznam virtuálních burz škol, ve kterém má každá burza přiřazený termín a lokalitu konání.

Střední školy se mohou registrovat k jednotlivým burzám a vložit odkaz na svou on-line konferenci. V den konání burzy se mohou deváťáci kliknutím na odkaz připojit do on-line konference vybrané školy.

Střední školy mají na portále založený svůj profil s informacemi o škole, seznam oborů a další informace jako informační brožura, či výsledky v soutěžích.

Odkaz na repositář s kódem: https://github.com/GrimirCZ/delta-burza.

Kapitola 1

Teoretická část

1.1 Serverová infrastruktura

Tato sekce popisuje prvky serverové infrastruktury, které jsou nasazeny v reálném řešení *BurzaŠkol.Online*.

1.1.1 Load Balancery

Load balancer[4] je prvek serverové infrastruktury určený k rozložení příchozích požadavků na skupinu backendových serverů. K rozložení požadavků je využíváno nejrůznějších algoritmů, jako například Round Robin, Least Connections, či IP Hash. Dělíme je na SW load balancery a HW load balancery.

HW load balancery jsou zpravidla proprietární a nákladná řešení. Vysokého výkonu dosahují využitím dedikovaných HW akcelerátorů. Jediné způsoby jejich škálování jsou zakoupení více fyzických strojů, či zakoupení silnějších strojů.

SW load balancery jsou distribuovány ve formě uživatelských aplikací spustitelných na běžném hardwaru, či ve virtuálních strojích cloudových providerů. Škálovatelnost je tedy velice triviální. V porovnání s HW load balancery stačí pouze on-demand zakoupit virtuální stroj s odpovídajícím výpočetním výkonem. Tato vlastnost činí z SW load balancerů velice flexibilní řešení, ideální pro agilní cloudové prostředí.

1.1.2 Objektová úložiště

Objektové úložiště je typ úložiště určený k ukládaní nestrukturovaných dat. Na rozdíl od tradičních způsobů ukládání dat, jakými jsou souborové systémy a bloková úložiště, pracují objektová úložiště[5] s daty jako se samostatnými objekty. Každý objekt obsahuje data samotná, uložená v jejich nativním formátu, a metadata. Metadata obsahují informace jako přístupová práva objektu, unikátní globální identifikátor a další přidružené informace, jako například formát uložených dat. Díky adresaci za pomocí globálních identifikátorů je možné objekty transparentně přesouvat. Tato vlastnost nám poskytuje nezávislost dat na použitém záznamovém médiu.

Je tedy například možné přesunout málo používané objekty z diskového úložiště na úložiště založené na magnetické pásce. Na druhou stranu ale také můžeme transparentně přesunout často požadované objekty z úložiště do pracovní paměti pro rychlý přístup.

Objektová úložiště dovolují ukládání velkého objemu nestrukturovaných dat. Je proto ideální jako úložiště pro uživatelské obrázky, dokumenty a zálohy. Díky své flexibilitě jsou objektová úložiště výrazně levnější než například bloková úložiště.

1.1.3 Relační databáze

Relační databáze[6] jsou úložiště strukturovaných dat, založená na tzv. relačním modelu. Relační model organizuje data do *tabulek*, *řádků* – záznamů a *sloupců*. Vzájemné vztahy mezi záznamy jsou reprezentovány *relacemi* – shodnými hodnotami dvou sloupců.

Software implementující relační databáze se nazývá Relational Database Management System (RDBMS). RDBMS dále obsahuje implementaci některého z dotazovacích jazyků, např. SQL, a podporu funkcí jako jsou ACID, triggery a uložené procedury.

SQL

Pro dotazování a správu dat uchovaných v RDBMS se využívá standardizovaný jazyk SQL (Structured Query Language)[7]. Díky němu můžeme data analyzovat a získávat z nich informace. Data můžeme shlukovat, transformovat a zkoumat pomocí nejrůznějších statistických funkcí.

Spravované databáze

Provize a správa databází je velice komplikovaná. Je třeba řešit replikaci, a také automatické škálování. Spravované databáze[8] se snaží tyto problémy řešit poskytováním předpřipravených databázových instancí, u kterých je automaticky řešena provize a škálování. Většina spravovaných databází dále poskytuje velice bezpečné nastavení vyladěné pro maximální bezpečnost a výkon databáze. To nám dovoluje se starat pouze o databázové objekty a uživatelské účty, tedy aspekty které přímo ovlivňují naši aplikaci.

Další výhodou spravovaných databází je poskytování různých metrik a varování, která nám ukazují detailnější informace o stavu a provozu databáze. Spravovaná databáze je například schopná detekovat tabulky bez primárního klíče a zaslat upozornění databázovému administrátorovi, či zobrazovat detailní real-time statistiky o propustnosti databáze.

1.1.4 Webové servery

Webový server (webserver)[9] je klíčový prvek infrastruktury webové aplikace. Hlavní rolí webserveru, jak už název napovídá, je poskytování webových stránek. Moderní webservery podporují širokou škálu protokolů jako je Hyper Text Transport Protocol (HTTP), Web Sockets (WS), či Secure Sockets Layer (SSL). Pro standardní internetovou komunikaci využívají webservery síťové porty 80 pro nešifrovanou komunikaci a 443 pro šifrovanou komunikaci.

Většina webserverů poskytuje obsah dvěma způsoby, staticky a dynamicky. Statický obsah je poskytován ze souborů uložených na webserveru samotném. Dynamický obsah je na druhou stranu generován samostatným

aplikačním serverem, který poté využívá webserver jako komunikační bránu. Některé webservery mají schopnost přímo provádět aplikační kód a separátní server není potřeba. Pro vytváření dynamických stránek můžeme například využít PHP, Perl, či různé CGI skripty. Statický obsah je ideální pro poskytování obsahu jako jsou obrázky, či klientské skripty. Dynamický obsah se na druhou stranu využívá pro obsluhu formulářů, či generování sestav.

1.2 Aplikační architektura

Tato sekce popisuje technologie využité při vývoji aplikace *BurzaŠkol.Online*.

1.2.1 HTTP

HTTP[10] je jeden z klíčových internetových protokolů. Mezi jeho hlavní přednosti patří jeho textová podoba a bezstavovost. Po síti tedy cestují pouze jednoduše stravitelné textové řetězce, převážně čitelné i pro člověka. Díky tomu je oproti ostatním přenosovým protokolům jednoduše použitelný a implementovatelný. Bezstavovost znamená, že jednotlivé HTTP požadavky na sebe nijak nenavazují a jsou plně nezávislé.

1.2.2 PHP

PHP[11] je skriptovací jazyk vytvořený v roce 1994 pro snadný vývoj dynamických a interaktivních webových aplikací. Hlavními přednostmi jazyka PHP je dynamický typový systém, extenzívní standardní knihovna a hluboká integrace s klíčovými webovými technologiemi. Funkcionalita jazyka PHP je dále rozšiřována velkým množstvím knihoven a frameworků. Pro jednoduchou správu těchto knihoven je využíván správce balíčků Composer[12].

1.2.3 PHP Frameworky

Vytváření rozsáhlé a propracované webové aplikace je těžký a dlouho trvající úkol. Z tohoto důvodu se využívají tzv. *frameworky*. Tyto frameworky obsa-

hují široké množství nástrojů pro usnadnění vývoje webových aplikací. Mezi nejčastěji poskytované nástroje patří ORM[13], směrovač[14], či šablonovací knihovna[15]. Dále může obsahovat nástroje na správu uživatelských sezení a různé další utility. Časté jsou i různé terminálové programy určené k ulehčení práce s daným frameworkem, např. pro generování kontrolerů, databázových migrací a správu mezipaměti.

Využitím frameworku můžeme významně zkrátit čas vývoje aplikace, jelikož autoři frameworku za nás učinil většinu architekturálních rozhodnutí a implementovali obslužný kód. Nám tedy zbývá pouze byznysová logika naší aplikace a nemusíme řešit věci jako je ukládání často využívaných dat do mezipaměti, či ověřování uživatelů.

Na druhou stranu využití frameworku sebou nese i jisté nebezpečí. Frameworky se mezi sebou liší, tudíž zkušenosti s jedním frameworkem nemusí být přenositelné na druhý. Často je tomu naopak - při používání frameworku nevhodným způsobem mohou vzniknout velké problémy, především po stránce výkonu aplikace. Dalším velkým nebezpečím mohou být bugy samotného frameworku, jejichž opravou, či obcházením ztrácíme čas určený pro vývoj aplikace samotné.

Laravel

Laravel[16] je PHP framework pro snadné tvoření komplexních webových aplikací. Hlavními výhodami frameworku Laravel je propracovaná, detailní dokumentace a obrovský ekosystém podpůrných knihoven, projektů a služeb, jako např. Laravel Vapor[17] nebo Laravel Nova[18].

1.2.4 Webové sokety

Většina webových aplikací vyžaduje dříve či později nějakou real-time funkcionalitu. Je mnoho způsobů, jak tuto funkcionalitu implementovat. Můžeme využít HTTP Polling[19] nebo Server-sent events (SSE)[20]. Velká slabina obou těchto přístupů je, že i když máme stále aktuální data ze serveru, tak

neexistuje způsob, jak tyto metody využít pro odesílání požadavků na server. Je proto nutné udělat separátní HTTP požadavek, což je velice neefektivní.

Protokol Web Sockets (WS)[21] se tento problém snaží řešit poskytováním plně duplexní komunikace mezi klientským programem a serverem. Jelikož se jedná o protokol cílený na použití ve webových prohlížečích, staví tento protokol nad protokolem HTTP, který používá pro navázání komunikace – handshaking. Využívá i stejné porty jako protokol HTTP, port 80 pro nešifrovanou a port 443 pro šifrovanou komunikaci.

Pusher

Pusher[22] je software stavějící nad WS protokolem poskytující pub-sub komunikaci v reálném čase mezi serverem a klientskými zařízeními. Je vhodný pro aplikace, které nemohou tyto funkcionality poskytovat samy o sobě, např. stránky napsané v PHP. Pro tyto aplikace poskytuje Pusher jednoduché HTTP rozhraní, skrze které je možné zprávy zasílat a docílit tím komunikace v reálném čase.

1.2.5 Datový model

Datový model[23] určuje způsob průtok dat, interakcí mezi daty a uložení dat aplikace. Od datového modelu se odvíjí většina aspektů aplikace, např. způsob fungování nebo rozložení formulářů a přehledů.

Návrh datového modelu je jedna z nejdůležitějších částí vývoje webové aplikace. Pokud pří návrhu uděláme chybu můžeme si vývoj aplikace velice ztížit. Na druhou stranu dobře připravený datový model velice zjednodušuje vývoj aplikace, jelikož může sloužit jako referenční manuál pro její rozložení.

1.3 Vývojové prostředí

Pro vývoj softwaru je zapotřebí specializovaných vývojových prostředí. Vývojová prostředí obsahují interpreter, či kompilátor (dle typu jazyka)[24], textový

editor a další pomocný software, např. linter[25] nebo jazykový server[26]. Hlavním účelem vývojových prostředí je umožnit vývoj softwaru a co nejvíce ho zjednodušit.

Jelikož je příprava vývojových prostředí náročný a zdlouhavý proces, najdeme řadu postupů, jak jej zjednodušit. Existují různé programy, které vývojové prostředí vytvoří a nastaví a snaží se při tom brát v potaz nejběžnější nastavení, která by běžný uživatel mohl potřebovat. Jedná se o programy jako *create-react-app*[27] a *Artisan Console*[28]. Tyto programy nabízejí funkcionality jako zakládání nového projektu, generování šablon pro různé kusy kódu a spouštění projektu.

Hlavní slabinou těchto programů je, že i když pomáhají s přípravou projektu, často nepomáhají se samotným vývojem. Tento problém se snaží řešit tzv. Integrovaná vývojová prostředí (IDE)[29].

1.3.1 Integrovaná vývojová prostředí (IDE)

Hlavním cílem IDE je pokrýt kompletní vývojový cyklus aplikace. Od založení projektu, přes psaní, až po její vydání. IDE proto obsahuje široké množství nástrojů. Na rozdíl od pomocných programů se ale jedná o ucelený balíček. Dále často obsahují editor kódu s inteligentním napovídáním a zvýrazňováním, jednu z hlavních výhod použití IDE. Samozřejmostí jsou také pokročilé možnosti ladění kódu s GUI debuggery nebo navigování pomocí symbolů.

Pokud by nám základní funkcionalita IDE nestačila můžeme využít dalších zásuvných modulů. Ty jsou poskytovány buďto autorem samotného IDE nebo třetí stranou. Zásuvné moduly poskytují funkcionality jako podporu dalších jazyků, či formátů souborů, rozšíření možností editování textu nebo integraci s dalšími službami.

Kapitola 2

Realizace projektu

2.1 Návrh datového modelu

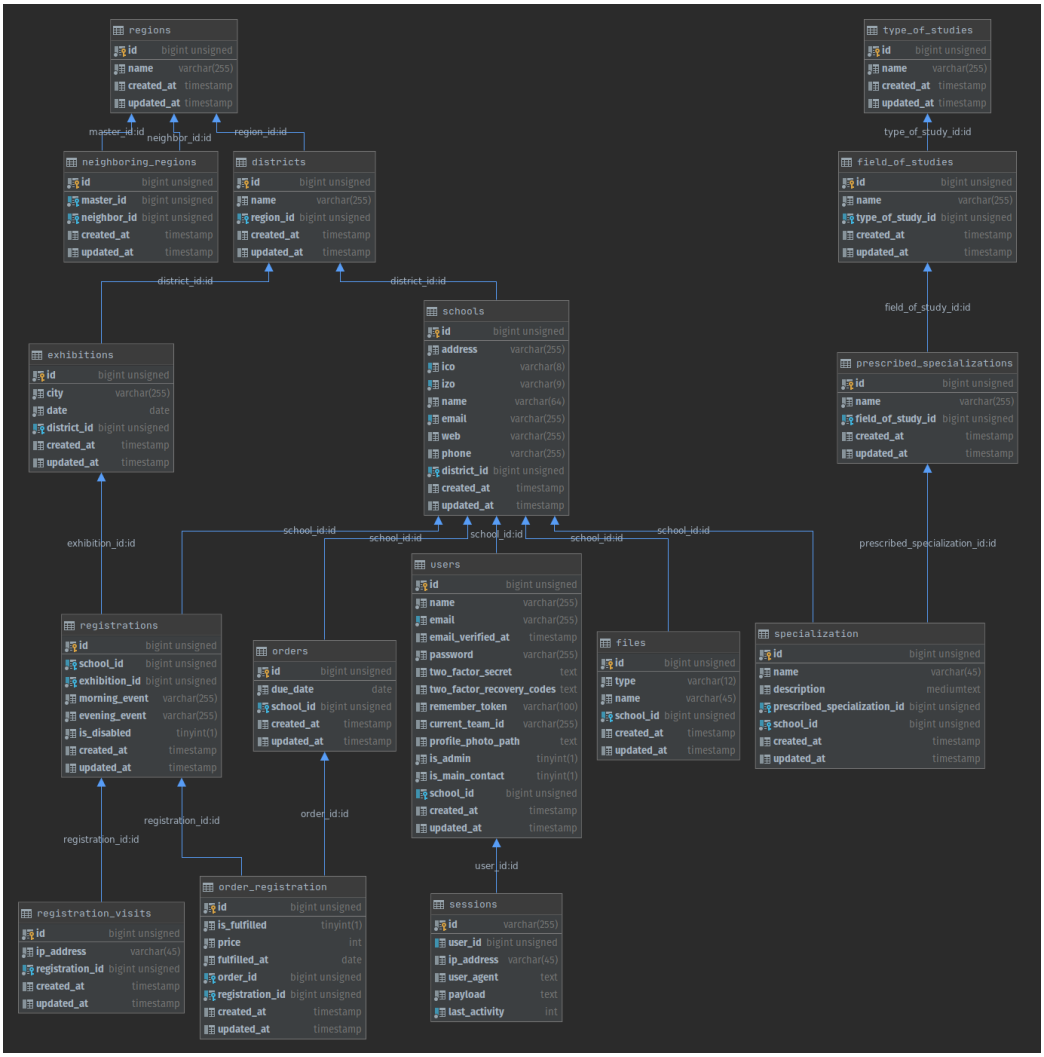
První krok[23] při návrhu datového modelu je vypracování zadání a přesné vymezení datových entit, zatím určujeme pouze název a účel entity. Následně musíme definovat relace mezi daty, tedy popsat, jakým způsobem mezi sebou data interagují.

Druhým krokem je vytyčení atributů datových entit. U atributu je třeba určit název, typ dat, která v něm budou uložena, např. datum. Je také nutné určit zda bude atribut držet unikátní hodnoty, např. emailové adresy a zda je atribut povinný nebo zda může nést prázdnou hodnotu, tzv. *null*[30].

Dalším krokem je adaptace tohoto obecného datového modelu, do modelu specifického pro prostředí, v kterém budou data uložena, např. relační databáze. V tomto kroku dojde k materializaci relací mezi entitami na cizí klíče a intermediate tabulky[31]. Podle atributů vytvořených v druhém kroku vytvoříme příslušné sloupce s odpovídajícími datovými typy a omezeními.

Posledním krokem, který nemusí nutně probíhat v době návrhu, ale může být proveden dodatečně, je vytipování často využívaných dat a vytvoření tzv. *indexů*[32]. Ty slouží pro optimalizaci databázových dotazů a urychlení vyhledávání dat. Tento krok je dobré provést až po vytyčení hlavních dotazů a identifikaci míst, která by indexováním mohla benefitovat. *Přílišné*

indexování může vést k nárůstu velikosti databáze a zpomalení dotazů![33]



Obrázek 2.1: Datový model aplikace *BurzaŠkol.Online*k říjnu roku 2020

Datový model *BurzaŠkol.Online*je postaven kolem konceptu vystavovatele, z implementačních důvodů reprezentováno tabulkou *schools*. Vystavovatelé jsou hlavní cílová skupina portálu. Mají možnost přihlásit se na výstavy a vložit krátký článek o sobě, případně o svých oborech, s podporou formátování za pomoci HTML značek. Portál umožňuje vystavovatelům nahrávat loga, informační brožury a obrázky, aby mohli zájemcům poskyt-

nout maximální množství informací. V současné podobě projektu *BurzaŠkol.Online*existují tři typy vystavovatelů: školy, firmy a Úřady Práce České Republiky.

Školy jsou hlavním typem vystavovatele. Mohou si vytvářet obory, které dále přiřazují k oborům z číselníku MŠMT. To nám dovoluje poskytovat filtrování škol podle typu studia, zaměření a oborů samotných. Školy jsou také automaticky párovány s výsledky maturitních zkoušek, výsledku soutěží zařazených do programu Excelence MŠMT a k inspekčním zprávám. Hlavní přínos portálu *BurzaŠkol.Online*pro školy je navázání kontaktu s žáky 9. tříd a propagace.

Druhým významným typem vystavovatele jsou firmy. Ty mohou vyjádřit svou podporu školám skrze funkcionalitu spolupráce. Kromě toho mají možnost seznámit zájemce se svou nabídkou pracovních pozic a stipendijních programů. Úřady práce mohou poskytovat nerozhodným žákům rady a dopomoci jim k vybrání vysněného studijního oboru.

2.2 Vývoj projektu

V této sekci jsou popsány technologie využité při vývoji portálu *BurzaŠkol.Online*a důvody pro jejich výběr, postup vývoje projektu a získané zkušenosti.

2.2.1 Využité technologie

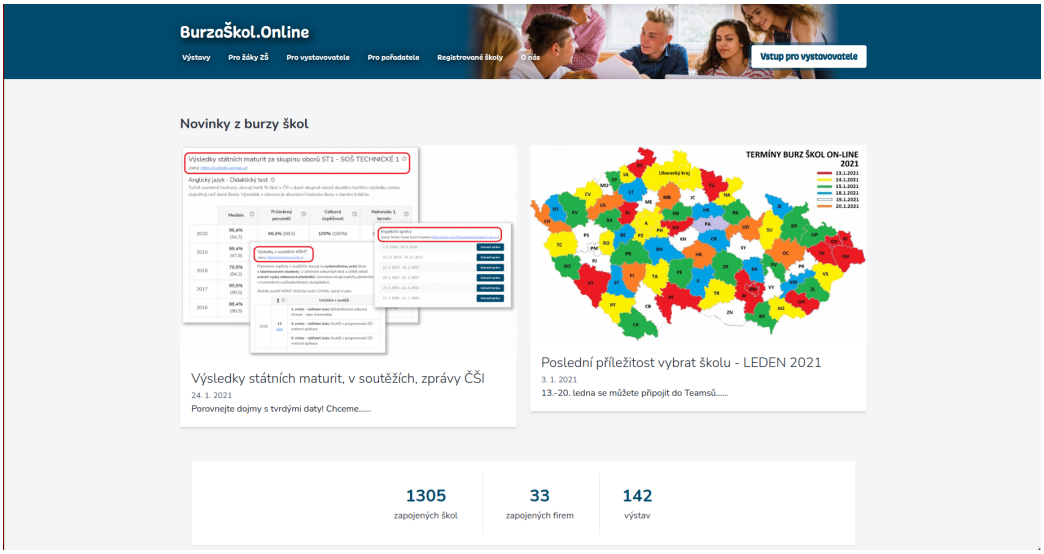
Při vývoji projektu byl využit programovací jazyk PHP s frameworkem Laravel. Pro real-time funkcionality, jako chat a notifikace, byl využit open-source implementace aplikace Pusher, Laravel Websockets[34]. Jako RDBMS jsem využil MySQL[35]. Emailové služby pro aplikaci zajišťuje služba Amazon SES[36].

PHP a Laravel

Jelikož byla na začátku projektu *BurzaŠkol.Online*klíčová rychlost dodání MVP byl vybrán jazyk PHP společně s frameworkem Laravel z důvodu jed-

noduchosti nasazení, nízkých nákladů na provoz a velké flexibility a rychlosti vývoje. Dalším velkým plus jsou pomocné utility jako Composer[12] nebo Laravel Artisan Console[28] a rozšíření Laravel Idea[37] pro IDE PhpStorm[38].

Laravel, skrze Laravel Jetstream[39], poskytuje předpřipravené části aplikace jako registrace a ověřování uživatelů, CSS framework Tailwind CSS[40] a JavaScriptový scaffolding Inertia.js[41] nebo Laravel Livewire[42].



Obrázek 2.2: Aplikace *BurzaŠkol.Online*

Pusher

Technologii Pusher byla vybrána z důvodu jednoduché integrace do stávajícího projektu a možnosti pokročilého ladění. Z implementačních důvodů byla využita open-source implementace protokolu Pusher nazvaná „Laravel Websockets“. To nám poskytlo větší kontrolu nad nasazením a škálováním projektu.

MySQL

RDBMS MySQL oproti ostatním relačním databázím vyniká jednoduchostí správy a množstvím poskytovatelů spravovaných databázových instancí[43].

To, a excelentní integrace s jazykem PHP a frameworkem Laravel, z ní dělá ideální volbu pro vývoj webových aplikací. Velkým přínosem je také široká řada nástrojů jako mysqldump[44] nebo phpMyAdmin[45], které významně zjednodušují správu a provoz databáze MySQL.

Amazon SES

Amazon Web Services (AWS) poskytuje v rámci služby Simple Email Service (SES) jednoduchou a cenově dostupnou emailovou bránu. Ta nabízí funkcionality jako DKIM podepisování emailů a zobrazování pokročilých statistik jako počet odeslaných emailů, počet „odražených“ emailů[46] a počet stížností na námi zaslanou poštu.

2.2.2 Přípravy na vývoj projektu

Předtím než může započít vývoj SW projektu je zapotřebí se zadavatelem vytvořit zadání. Jedná se o dokument popisující jednotlivé konkrétní funkcionality a procesy aplikace. Ze zadání by měly vycházet veškeré části aplikace jako je datový model a použité technologie. *Zadání by mělo být konkrétní a jednoznačné, pokud existuje více výkladů, může dojít k velkým problémům při vývoji aplikace.*

Dalším krokem je výběr technologií, které budou na vývoj projektu použity. Ze zadání určíme pro jakou platformu bude projektu určen, zda-li se jedná o webovou, desktopovou, mobilní nebo serverovou aplikaci. Dle toho vybereme konkrétní technologie jako např Laravel[16] pro vývoj aplikace, MySQL[35] pro ukládání dat a Git[47] pro správu zdrojového kódu.

Když jsou zvoleny technologie můžeme začít vytvářet datový model pro vybraný typ databáze. Dále rozvrhneme jednotlivé obrazovky a postupy, které bude aplikace obsahovat. Je také potřeba vytvořit plán nasazení projektu, tedy jak bude aplikace hostováná, či jaký mail server využijeme.

Když máme všechny přípravy hotovy můžeme přejít k samotnému programování projektu.

2.2.3 Vývoj projektu ve frameworku Laravel

Prvním krokem při vývoji projektu ve frameworku Laravel, je vytvoření tříd tzv. *modelů*[48] a k nim náležejících *migrací*[49]. Ty vytvoříme dle připraveného datového modelu.

Když jsou modely a migrace úspěšně vytvořeny můžeme přistoupit k vytváření *kontrolerů*[50], *komponent*[51] a definování *cest*[52].

Modely

Modely jsou PHP třídy reprezentující samostatné databázové tabulky, řádky a relace mezi nimi. Modely nám dovolují vytvářet, upravovat a filtrovat záznamy jako by se jednalo o nativní PHP objekty. To velice zjednodušuje práci s daty.

V současné době aplikace *BurzaŠkol.Online*obsahuje 26 modelů, např. *School* pro vystavovatele nebo *Region* pro kraj.

Migrace

Migrace, ve frameworku Laravel, slouží pro kontrolované úpravy databáze. Jejich hlavní úkol je udržet databázi v konzistentním stavu s verzí aplikace. Při nasazení nové verze tedy stačí spustit migrace a databáze je synchronizována s verzí aplikace.

HTTP cesty

HTTP cesty definují jaký kód je spuštěn při HTTP požadavku. Cesta je identifikována pomocí jedné, či více HTTP metod a HTTP cesty. Cesta v sobě může obsahovat parametry, např. id knihy. Cesta může vést na *HTTP přesměrování*, *PHP funkci* nebo *HTTP kontroler*.

Aplikace *BurzaŠkol.Online*definuje 38 unikátních HTTP cest. Velké množství HTTP cest dále přijmá různé parametry, které dále ovlivňují obsah vygenerované stránky.

HTTP kontrolery

HTTP kontrolery, ve frameworku Laravel, jsou PHP třídy obsahující logiku pro zpracování HTTP cest – handlery, PHP metody. Můžeme je rozdělit na standardní kontrolery, *single-action kontrolery*[53] a *resource kontrolery*[54].

Standardní kontrolery obsahují libovolný počet handlerů. Při definování HTTP cesty musíme definovat jak kontroler, tak metodu, která má být zavolána.

Single-action kontrolery obsahují pouze metodu *__invoke*. Jsou vhodné pokud je logika pro nějakou cestu velice komplexní a je lepší ji izolovat. Při definování cesty stačí zmínit pouze název třídy a Laravel metodu automaticky spáruje.

Resource kontrolery mají přesně definovanou strukturu. Obsahují skupinu metod pro vytváření, úpravu a čtení daného zdroje – resource. Cesty se definují skupinově pomocí statické metody *resource*.

Blade komponenty

Všechny aplikace mají za cíl zobrazovat uživateli informace. Webové aplikace využívají k zobrazování informací značkovací jazyk HTML.

U jednoduchý stránek můžeme využít statických HTML souborů, které jsou pro každý požadavek stejné a nemění se. Pro úpravu obsahu stránky je nutný přístup k hostingové službě a znalost HTML.

Moderní webové aplikace vyžadují obsah dynamický, který je závislý na velké řadě faktorů, např. zda-li je uživatel přihlášen, jaká jsou jeho oprávnění, či jaké výstavy se právě konají. Obsah těchto stránek je většinou uživatelsky editovatelný za pomocí grafického WYSIWYG editoru.

Pro generování těchto stránek se používají šablonovací nástroje jako samotné PHP nebo různé pomocné knihovny jako Twig[55] nebo Laravel Blade Views[56]. Takové knihovny nám dovolují psát čisté HTML s přídavkem speciálních příkazů. Tyto příkazy nám dovolují například podmínečné generování, možnost vypisovat a formátovat PHP proměnné nebo modularizaci stránek na menší znovupoužitelné komponenty.

2.3 Nasazení a provoz projektu

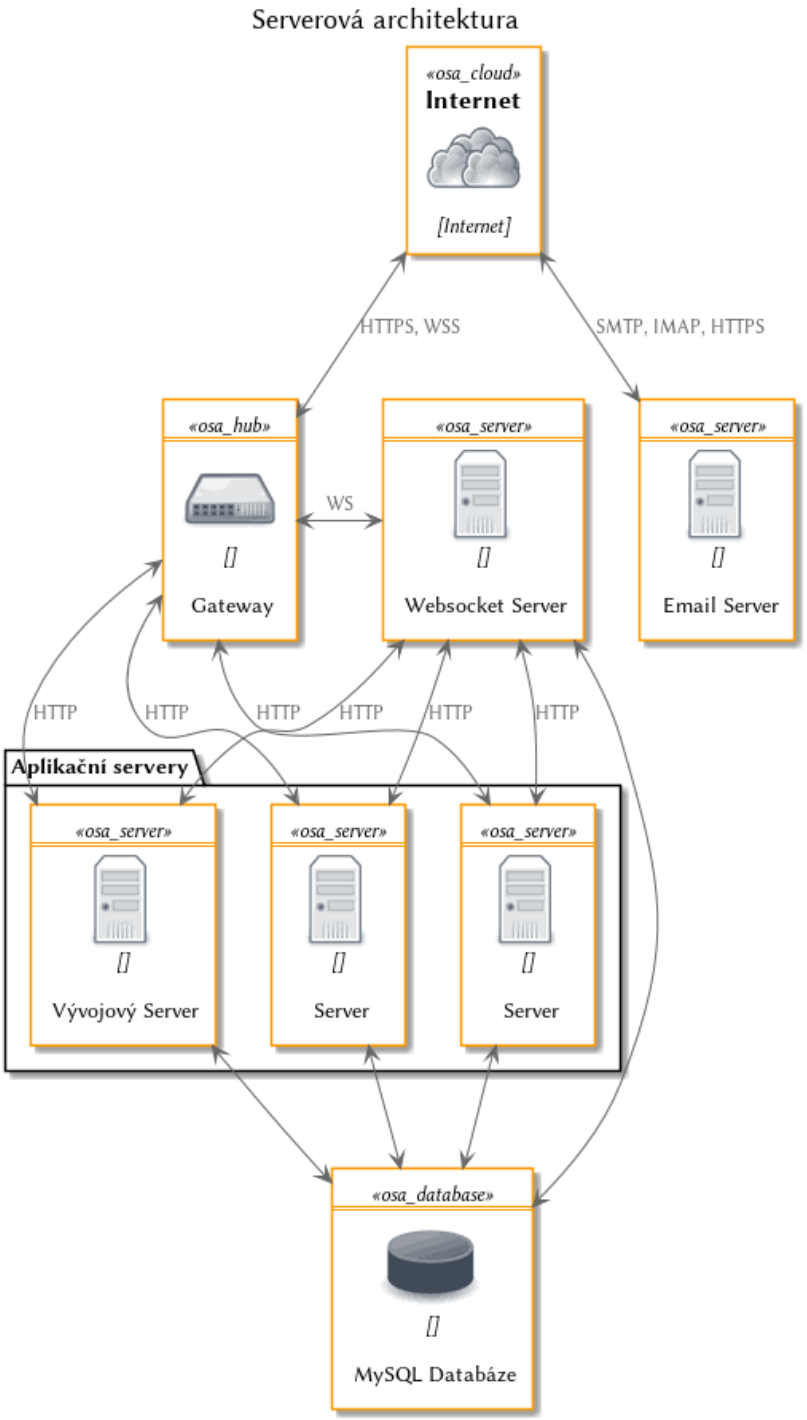
Tato kapitola je věnována nasazení a provozu projektu *BurzaŠkol.Online*. Popisuje volbu hostingové služby a důvody k ní vedoucí, serverovou architekturu a obsahuje také poznatky a překonané problémy, které v průběhu nasazování projektu vznikli.

2.3.1 Hostingová služba

Při navrhování projektu *BurzaŠkol.Online*byly zvažovány služby Amazon AWS[57], Microsoft Azure[58] a cloudem společnosti Digitalocean[59]. Všechny tři služby poskytují velké množství pokročilých funkcí pro vývoj a provoz webových aplikací.

Nakonec byl pro většinu funkcí zvolen cloud společnosti Digitalocean, jelikož má ze všech tří poskytovatelů nejjednodušší správu a finanční politiku, je přesně uvedeno za co platíte a neexistují žádné skryté limity a kvóty. Mezi další výhody cloudu Digitalocean patří vysoký výkon a velmi příznivé ceny.[60]

Pro další funkcionality jako ukládání souborů a zasílání emailů byly využity služby Amazon S3[61] a Amazon SES[36]. Důvody pro toto rozhodnutí byly bezkonkurenční ceny a integrace s použitými technologiemi, hlavně frameworkem Laravel.



Obrázek 2.3: Serverová architektura projektu *BurzaŠkol.Online*

2.3.2 Serverová architektura

> Internet je divokým a nebezpečným místem[62].
>
> Software a Hardware je nespolehlivý, chyba není výjimka ale pravidlo[63].

Podle těchto dvou faktů musíme řídit svůj návrh serverové infrastruktury. Musíme tedy vytvořit serverovou infrastrukturu, která je odolná, bezpečná a škálovatelná.

Z hlediska bezpečnosti můžeme rozdělit servery projektu *BurzaŠkol.Online*na dvě skupiny: okrajové servery a interní servery.

Okrajové servery mají za úkol komunikaci s Internetem. Jedná se o několik přesně definovaných míst, skrze něž prochází všechna komunikace určená mimo naši infrastrukturu.

Jednou z hlavních výhod tohoto přístupu je možnost využívat uvnitř naší infrastruktury nešifrované protokoly jako HTTP a provádět jejich zašifrování pouze na těchto dedikovaných okrajových serverech. Tyto nešifrované protokoly jsou výkonnější a jednodušší než jejich šifrované varianty[64]. Z důvodu bezpečnosti je ale není možné použít pro Internetovou komunikaci.

Z pohledu resilience, obrany proti výpadku, jsou nejdůležitějšími opatřeními replikace a load-balancing klíčových částí. Využíváme proto load balancery a spravované databáze, u kterých neznamená výpadek jednoho stroje nedostupnost celé aplikace.

2.3.3 Postup nasazení aplikace

Prvním krokem je provize serveru. Jelikož projekt *BurzaŠkol.Online*obsahuje pouze malý počet serverů, bylo možné všechny servery nakonfigurovat a vytvořit přímo z ovládacího panelu Digitalocean.

Poté co jsou servery vytvořeny přejdeme ke konfiguraci samostatných serverových rolí. Každá serverová role vyžaduje separátní konfiguraci a specifický software, aby mohla svou funkci vykonávat.

Dalším krokem je vytvoření obrazů disků nakonfigurovaných serverů, abych mohl rychle a jednoduše vytvářet další servery těchto rolí. Pro každou serverovou roli byl vytvořen seznam parametrů, které je po vytvoření serveru z obrazu disku nutné upravit. U aplikačních serverů je například nutné upravit IP adresu, na které má webový server naslouchat pro nová připojení.

Posledním krokem je nastavení příslušných DNS záznamů a získání SSL certifikátů.

2.3.4 Nasazování aktualizací aplikace

První přístup nasazování aktualizací, který byl na nasazování projektu *BurzaŠkol.Online* využit, byl založený na Github Actions[65]. Github Actions je CI/CD mechanismus založený na spouštění kódu v závislosti na událostech verzovacího systému Git[47]. Využíván byl SSH for Github Actions[66], pro automatické nasazování při commitu do dedikované větve.

Tento přístup má několik problémů. Hlavním problémem je fakt, že pro každý přidaný cílový server je zapotřebí vytvořit novou konfiguraci. Když vysazovací akce selže, je těžké najít příčinu, jelikož výpisy Github Actions jsou nepřehledné a zaměňují STDOUT a STDERR. Posledním velkým problémem tohoto přístupu je, že pro spuštění vysazení je zapotřebí změna v kódu aplikace.

Tyto problémy jsem vyřešil použitím Laravel Envoy[67]. Laravel Envoy využívá konfigurační soubory v jazyce PHP pro provádění dávek akcí na skupině serverů zároveň. Pokud je zapotřebí přidat nový server, stačí přidat jeho IP adresu do seznamu serverů a Laravel Envoy automaticky provede všechny potřebné akce. Další velkou výhodou je absence nutnosti ukládat SSH klíče na serveru třetí strany. Pro zopakování vysazení stačí pouze znovu spustit Laravel Envoy.

2.4 Postup projektu

2.4.1 Jedno-serverové nasazení

První verze aplikace *BurzaŠkol.Online* využívala pouze jeden server pro hostování všech aplikačních funkcí jako webserver, databázový server a aplikační server. Tento způsob hostování má však mnoho problémů.

Jelikož celá infrastruktura běží na jednom serveru narážíme na velké bezpečnostní riziko vytvářením kritického bodu infrastruktury, jehož nedostupnost nebo porucha znamená výpadek celé aplikace. To také znamená, že není možné aplikaci aktualizovat bez jejího výpadku.

Druhým, i když menším problémem, je omezení možnosti škálování v odpovědi na nárůst požadavků, jelikož pro alokaci větších serverových prostředků je nutné server vypnout. Pro aplikaci *BurzaŠkol.Online* toto nepředstavovalo velký problém jelikož jazyk PHP je velice efektivní ve využívání serverových prostředků a provoz aplikace nepřekročil alokované serverové prostředky.

2.4.2 Více-serverové nasazení

Pro vyřešení tohoto problému byla infrastruktura byla rozšířena na více serverů.

Jako vstupní bod infrastruktury byl vyčleněn server označen *gateway*. Mezi hlavní role tohoto serveru byly určeny load-balancer a vstupní bod protokolů HTTP a WS. To nám dovoluje provádět provizi ssl certifikátů a ssl terminaci pouze v jednom místě naší infrastruktury.

Jelikož *gateway* poskytuje služby load-balanceru mohli jsme vytvořit několik, na sobě nezávyslých, instancí aplikace *BurzaŠkol.Online*. Pro provedení tohoto kroku bylo zapotřebí rozčlenit všechny zdroje dat, např. RDBMS nebo ukládání souborů, na dedikované servery, jelikož aplikační servery nemohou tyto služby mezi sebou sdílet. Jako RDBMS jsem využil spravovanou databázy společnosti Digitalocean, která poskytuje replikaci a redundanci databázových serverů. Pro ukládání souborů jsem využil objektové úložiště

Amazon S3, to nám dovoluje manipulovat se soubory ze všech aplikačních serverů najednou, bez potřeby vlastní infrastruktury.

Rozčlenění aplikačních serverů nám poskytuje větší odolnost proti výpadku samostatných serverů, výpadek serveru neznamená výpadek aplikace, pouze zhoršení dostupnosti. Také nám dovoluje aplikaci škálovat bez potřeby její odstávky.

2.4.3 Provoz aplikace

Ve špičce provozu aplikace *BurzaŠkol.Online* využíval load-balancer asi 30Mb/S síťového provozu. Systém v tu dobu navštívilo asi 790 návštěvníků, tj. kolem 25 návštěvníků na školu[1].

2.5 Přínos aplikace

2.6 Bezpečnost aplikace

Kapitola zaměřená na zabezpečení projektu *BurzaŠkol.Online*je z důvodu komplexity tématu rozdělena na dvě části. První část pojednává o zabezpečení systému proti útokům vedeným skrze samotnou aplikaci, např. útokům typu *SQL Injection*. Druhá kapitola se zaměří na zabezpečení infrastruktury proti útokům vedeným z internetu cílících na špatně nakonfigurované služby, např. *Útok na SSH přihlašování serveru*.

2.6.1 Aplikační zabezpečení

Hlavní nebezpečí při provozování Webového portálu, plyne z práce s uživatelským vstupem. Aplikace musí tento vstup přijímat a operovat nad ním. To může vést k celé řadě útoků. Je tedy důležité, aby uživatel svým vstupem nemohl

[1]Čísla nejsou exaktní z důvodu nemožnosti přesného měření návštěvnosti webových stránek.

webovou aplikaci napadnout, zpomalit, ani jinak vyřadit z provozu. Nejdříve se zaměříme na bezpečnost textových polí a útoky s nimi spojenými.

SQL Injection

SQL Injection[68] je typ útoku, při kterém útočník zadá do textového vstupu kód v jazyce SQL. Naše aplikace poté tento kód vezme a přímo ho vloží do řetězce dotazu.Při vykonávání příkazu pak databázový server vykoná tento škodlivý kód, jako by pocházel od samotné aplikace.

Tento útok představuje velké nebezpečí, jelikož může vést od ztráty dat, až k úniku citlivých dat do rukou útočníka.

Aplikace *BurzaŠkol.Online*se proti tomuto typu útoku brání pomocí *předpřipravených dotazů*[69]. Jedná se o techniku, kdy je databázový dotaz rozdělen na dvě části. První část tvoří samotné příkazy, které popisují, jak bude daný dotaz proveden. Druhou část tvoří uživatelské vstupy. Tato část obsahuje pouze data a databázový server ji nikdy nespouští. Tím zabraňuje tomu, aby server vykonal uživatelský vstup jako příkaz. *Předpřipravené dotazy*[69] riziko tohoto útoku plně eliminují.

Cross-site skriptování

Cross-site skriptování[70], nebo také XSS, je typ útoku, při kterém je do textového vstupu zadán validní kód spustitelný prohlížečem. Zranitelný webserver poté tento kód vloží přímo do stránky, která je zaslána na klientské zařízení. Klientské zařízení tento škodlivý kód následně vykoná v domnění, že se jedná o kód naší aplikace.

Tento útok představuje pro návštěvníky velké nebezpečí, jelikož může vést ke krádeži uživatelských přihlášení, zobrazování reklam nebo krádeži platebních údajů. Mezi stránkové skriptování[70] může být také využito jako mezikrok pro další, např. phishingové[71], útoky.

Potenciálními místy, kde by mohlo u aplikace *BurzaŠkol.Online*k napadení XSS dojít, jsou jednořádková vstupní pole a grafické textové editory. Jednořádková vstupní pole slouží pro jednoduchý uživatelský vstup bez

formátování. Zabezpečení je tedy velice jednoduché. Jakýkoliv vstup, který uživatel zadá, je očištěn a veškeré HTML značky jsou odstraněny. U grafických textových editorů však není tato obrana možná, jelikož validním vstupem je i HTML kód. Aplikace tedy daný kód na serveru parsuje a odstraní veškeré závadné HTML značky, jako je například značka *script*, a závadné HTML atributy, jako například *onmove*.

Podvržení požadavků mezi stránkami

Podvržení požadavků mezi stránkami (CSRF) [72] je typ útoku, při kterém cizí stránka obsahuje odkaz, který vede na naši webovou adresu. Většinou se jedná o požadavky typu POST. Tento požadavek má většinou za cíl vykonat škodlivou akci, např. napsat příspěvek bez vědomí uživatele. Při útoku je využíváno faktu, že prohlížeč s každým požadavkem odesílá HTTP cookies, které ověřují uživatele serveru.

Útočník tedy může například vytvořit novou objednávku a tím způsobit finanční újmu. Tento útok představuje pro uživatele velké nebezpečí.

Aplikace *BurzaŠkol.Online* se brání podvržení požadavků tak, že pro každou uživatelskou akci vyžaduje tzv. CSRF Token[72]. Token je unikátní pro každý požadavek a dovoluje uživateli vykonat přesně jednu akci. Token je vygenerován serverem, přímo do HTML kódu stránky. Útočník k němu proto nemá přístup a není mu dovoleno vykonat žádnou akci.

Útok hrubou silou

Útoky hrubou silou[73] je skupina útoků využívající velkého počtu požadavků za účelem uhodnutí uživatelských přihlašovacích údajů. Tyto požadavky jsou vedeny na přihlašovací formulář aplikace a podle odpovědi serveru určují zda byl útok úspěšný, či nikoliv.

Útok typu brute-force vede k vyzrazení přihlašovacích údajů útočníkovi. Útočník tím získá možnost plné kontroly nad uživatelským účtem.

Proti tomuto útoku je nasazena obrana spočívající v limitaci počtu požadavků na IP adresu. Pokud uživatel udělá za určený časový více požadavků na

přihlášení než je povolený limit, tak aplikace požadavky zahazuje a dále je nezpracovává. Tato ochrana činí útoky typu brute-force velice nepraktické, až nemožné.

2.6.2 Zabezpečení infrastruktury

Dalším důležitým faktorem bezpečnosti, při provozování webového portálu, je zabezpečení serverové infrastruktury. Útoky na serverovou infrastrukturu aplikace mají často větší dopad než zranitelnosti aplikace samotné. Tyto útoky mají často za úkol aplikaci vyřadit z provozu nebo ji zneužít. Jedná se, např. o *slovníkové útoky nebo bruteforce útoky na SSH přihlašování* nebo o *denial of service* útoky.

Útoky na SSH přihlašování

Při útoku na SSH přihlašování serveru se využívá povoleného přihlašování pomocí uživatelských hesel. Útok se snaží najít kombinace uživatelských jmen a hesel, které nejsou dostatečně komplexní a bezpečné. Častým cílem útoku je kořenový uživatel, *root*, který by úspěšnému útočníkovi poskytl kompletní kontrolu nad cílovým strojem. K prolomení hesel se využívají tzv. slovníkové útoky - soubory velkého množství častých hesel, kterými se útočník snaží zabezpečení prolomit.

Pokud dojde k prolomení SSH přihlášení serveru získá tím útočník plný přístup k serverovému terminálu. Je tedy schopný vykonávat všechny akce, ke kterým má napadený účet oprávnění. Útočník může například zapojit stroj do botnetu[74], zfalšovat nebo zaměnit poskytované stránky, či ukrást platební nebo osobní údaje.

Jednou z hlavních obran proti tomuto útoku je zákaz používaní hesel pro vzdálený přístup. K přihlašování se poté využívá tzv. SSH klíčů[75] - certifikátů založených na asymetrické kryptografii. Druhou linií obrany je poté rozřazení jednotlivých úloh pod samostatné uživatelské účty a princip minimálních oprávnění. Toto opatření pak minimalizuje škody způsobené útočníkem na úzkou napadenou oblast.

Denial of service útoky

Útoky typu Denial of Service (DoS)[76] jsou útoky při kterých se útočník snaží zahltit naši infrastrukturu, a tím způsobit její výpadek. Nejčastější variantou DoS útoku je Distributed Denial of Service (DDoS) - DoS útok vedený z mnoha míst najednou [77]. Vyskytují se ve velkém množství variací, např. Slowloris, různé typy floodingu a amplifikační útoky.

Slowloris útoky

DoS útok Slowloris[78] využívá toho, že každé připojení vyžaduje, tzv. File Descriptor (FD)[79]. FD je systémový zdroj, který v unixových systémech reprezentuje otevřené soubory, či síťová připojení. Každý běžící proces má omezený počet FD, které je schopný otevřít v jednu chvíli. Cílem útoku Slowloris je donutit webserver k vyčerpání počtu dostupných FD tím, že otevře velkou spoustu připojení a zasílá pouze tolik dat, aby server připojení neuzavřel. Snaží se tak napodobit spoustu zařízení s pomalým připojením. Když dojde k vyčerpání volných FD ztratí server schopnost přijímat nové požadavky a na koncového uživatele působí jako přetížený. Hlavní výhoda tohoto útoku je, že nevyžaduje větší prostředky ke svému provedení. Hlavní obranu proti útoku tvoří omezení maximálního množství připojení na IP adresu. Je také nutné nastavit rozumnou spodní hranici rychlosti připojení.

Floodingové útoky

Floodingové DDoS útoky jsou založené na přetížení cílového stroje. Jejich cílem je překročení maximální propustnosti HW a SW komponent serveru, a tím způsobit nedostupnost, výpadek, či dokonce fyzické poškození. Tento útok využívá nedokonalosti různých protokolů a jejich implementací. Příkladem floodingových útoků je UDP flood[80], SYN flood[81] nebo HTTP flood[82].

Amplifikační útoky

Amplifikační DDoS útoky jsou variantou floodingových útoků využívající základní internetové služby, jejichž velikost odpovědi je několikanásobně větší

než velikost požadavku. U některých dotazů je poměr až 1:200 [83]. Hlavními cíli amplifikačních útoků jsou nechráněné NTP servery[83] a DNS servery[84]. Při útoku se využívá faktu, že některé NTP a DNS servery přijímají požadavky pomocí UDP protokolu, bez kontroly pravosti zpáteční IP adresy. Útočník zašle malý UDP požadavek s podvrženou IP adresou cílového serveru. Zranitelný server poté odpoví na podvrženou IP adresu a provede UDP flood útok pod záminkou odpovědi na legitimní dotaz.

Obrana proti floodingovým a amplifikačním útoků

Obrana proti floodingovým a amplifikačním útokům je velice náročná. Jeden z největších problémů je fakt, že škodlivý datový provoz je v mnohých případech těžce rozeznatelný od normálního datového provozu aplikace. Je proto potřeba izolovat útočící IP adresy od IP adres legitimních uživatelů. Problém s blokací škodlivého provozu těchto útoků je, že samotný objem blokovaných dat má potenciál naši infrastrukturu přetížit. Ideálním, i když nákladným, řešením jsou fyzické firewally, které upravují provoz pomocí hardwarových akcelerátorů. Méně efektivními, ale stále užitečnými, možnostmi je alokace větších serverových prostředků na místa, kde naše služby komunikují s veřejným internetem, v kombinaci s blokováním útočících IP adres.

Závěr

V rámci práce byl vytvořen portál, který umožňuje nahrazení prezenčních Burz škol on-line video konferencemi v době pandemie koronaviru Covid-19.

Portál byl navržen, vyvinut, nasazen, je používán a vzhledem k současné pandemické situaci má potenciál zůstat důležitým prvkem i do budoucna.

TODO: Rozvést

Seznamy

Literatura

Literatura

1. *Microsoft Teams* [online]. [B.r.] [cit. 2021-02-21]. Dostupné z: https://www.microsoft.com/en/microsoft-teams/group-chat-software.
2. *Google Meet* [online]. [B.r.] [cit. 2021-02-21]. Dostupné z: https://meet.google.com/.
3. *Zoom* [online]. [B.r.] [cit. 2021-02-21]. Dostupné z: https://zoom.us/.
4. *What Is Load Balancing?* [Online] [cit. 2021-02-01]. Dostupné z: https://www.nginx.com/resources/glossary/load-balancing/.
5. *What is Cloud Object Storage?* [Online] [cit. 2021-02-01]. Dostupné z: https://aws.amazon.com/what-is-cloud-object-storage/.
6. *Co je relační databáze* [online] [cit. 2021-02-02]. Dostupné z: https://www.oracle.com/cz/database/what-is-a-relational-database/.
7. HELLER, Martin. *What is SQL? The lingua franca of data analysis* [online] [cit. 2021-02-02]. Dostupné z: https://www.infoworld.com/article/3219795/what-is-sql-the-lingua-franca-of-data-analysis.html.
8. DRAKE, Mark. *What is SQL? The lingua franca of data analysis* [online] [cit. 2021-02-03]. Dostupné z: https://www.digitalocean.com/community/tutorials/understanding-managed-databases.
9. *What is a web server?* [Online] [cit. 2021-02-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server.

10. *HTTP* [online] [cit. 2021-02-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP.

11. *PHP* [online]. [B.r.] [cit. 2021-02-04]. Dostupné z: https://www.php.net/.

12. NILS ADERMANN, Jordi Boggiano; CONTRIBUTORS, community. *Composer* [online]. [B.r.]. Ver. 2.0.9 [cit. 2021-02-04]. Dostupné z: https://getcomposer.org/.

13. HOYOS, Mario. *What is an ORM and Why You Should Use it* [online] [cit. 2021-02-04]. Dostupné z: https://blog.bitsrc.io/what-is-an-orm-and-why-you-should-use-it-b2b6f75f5e2a.

14. PAUL, John O. *How to build a basic server side routing system in PHP.* [Online] [cit. 2021-02-04]. Dostupné z: https://medium.com/the-andela-way/how-to-build-a-basic-server-side-routing-system-in-php-e52e613cf241.

15. UNDERWOOD, Paul. *The Ups and Downs of PHP Template Engines* [online] [cit. 2021-02-04]. Dostupné z: https://dzone.com/articles/ups-and-downs-php-template.

16. *Laravel* [online]. [B.r.] [cit. 2021-02-08]. Dostupné z: https://laravel.com/.

17. *Laravel Vapor* [online]. [B.r.] [cit. 2021-02-08]. Dostupné z: https://vapor.laravel.com/.

18. *Laravel Nova* [online]. [B.r.] [cit. 2021-02-08]. Dostupné z: https://nova.laravel.com/.

19. HANSON, Joe. *What is HTTP Long Polling?* [Online] [cit. 2021-02-08]. Dostupné z: https://www.pubnub.com/blog/http-long-polling/.

20. *Using server-sent events* [online] [cit. 2021-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events/Using_server-sent_events.

21. *The WebSocket API (WebSockets)* [online] [cit. 2021-02-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API.

22. *Pusher* [online]. [B.r.] [cit. 2021-02-08]. Dostupné z: https://pusher.com/.

23. *DBMS - Data Models* [online] [cit. 2021-02-09]. Dostupné z: https://www.tutorialspoint.com/dbms/dbms_data_models.htm.

24. *Compiler vs Interpreter* [online] [cit. 2021-02-06]. Dostupné z: https://www.geeksforgeeks.org/compiler-vs-interpreter-2/.

25. *What is a linter and why your team should use it?* [Online] [cit. 2021-02-06]. Dostupné z: https://sourcelevel.io/blog/what-is-a-linter-and-why-your-team-should-use-it.

26. *Language Server Protocol* [online] [cit. 2021-02-06]. Dostupné z: https://microsoft.github.io/language-server-protocol/.

27. EAN PLATTER Jonny Buchanan, Max Stoiber. *Create React App* [online]. [B.r.] [cit. 2021-02-06]. Dostupné z: https://github.com/facebook/create-react-app.

28. *Artisan Console* [online]. [B.r.] [cit. 2021-02-06]. Dostupné z: https://laravel.com/docs/8.x/artisan.

29. *Programming software and the IDE* [online] [cit. 2021-02-06]. Dostupné z: https://www.bbc.co.uk/bitesize/guides/zgmpr82/revision/1.

30. *What is a Database NULL Value?* [Online] [cit. 2021-02-09]. Dostupné z: https://www.essentialsql.com/get-ready-to-learn-sql-server-what-is-a-null-value.

31. BRADLEY, Paul. *KB5772: What is an "intermediate table"?* [Online] [cit. 2021-02-09]. Dostupné z: https://community.microstrategy.com/s/article/KB5772-What-is-an-quot-intermediate-table-quot.

32. *Database Indexes Explained* [online] [cit. 2021-02-09]. Dostupné z: https://www.essentialsql.com/what-is-a-database-index/.

33. OZAR, Brent. *Index Tuning Week: How Many Indexes Are Too Many?* [Online] [cit. 2021-02-09]. Dostupné z: https://www.brentozar.com/archive/2018/10/index-tuning-week-how-many-indexes-are-too-many.

34. MARCEL POCIOT Freek Van der Herten, Community Contributors. *Laravel Websockets* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://github.com/beyondcode/laravel-websockets.

35. *MySQL* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://www.mysql.com/.

36. *Amazon Simple Email Service (SES)* [online]. [B.r.] [cit. 2021-02-12]. Dostupné z: https://aws.amazon.com/ses/.

37. FAIZRAKHMANOV, Adel. *Laravel Idea* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://plugins.jetbrains.com/plugin/13441-laravel-idea.

38. *PhpStorm* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://www.jetbrains.com/phpstorm/.

39. *Laravel Jetstream* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://github.com/laravel/jetstream.

40. *Tailwind CSS* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://tailwindcss.com/.

41. *Inertia.js* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://github.com/inertiajs/inertia.

42. *Laravel Livewire* [online]. [B.r.] [cit. 2021-02-11]. Dostupné z: https://laravel-livewire.com/.

43. JANÍK, David. *Velké srovnání MySQL (MariaDB) a PostgreSQL* [online] [cit. 2021-02-12]. Dostupné z: https://www.vas-hosting.cz/blog-velke-srovnani-mysql-mariadb-a-postgresql.

44. *mysqldump* [online]. [B.r.] [cit. 2021-02-12]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/mysqldump.html.

45. *phpMyAdmin* [online]. [B.r.] [cit. 2021-02-12]. Dostupné z: https://www.phpmyadmin.net/.

46. *Soft vs. Hard Bounces* [online] [cit. 2021-02-12]. Dostupné z: https://mailchimp.com/help/soft-vs-hard-bounces/.

47. LINUS TORVALDS, Junio C Hamano. *Git* [online]. [B.r.] [cit. 2021-02-13]. Dostupné z: https://git-scm.com/.

48. *Eloquent: Getting Started* [online] [cit. 2021-02-14]. Dostupné z: https://laravel.com/docs/8.x/eloquent.

49. *Database: Migrations* [online] [cit. 2021-02-14]. Dostupné z: https://laravel.com/docs/8.x/migrations.

50. *Controllers* [online] [cit. 2021-02-14]. Dostupné z: https://laravel.com/docs/8.x/migrations.

51. *Blade Templates* [online] [cit. 2021-02-14]. Dostupné z: https://laravel.com/docs/8.x/blade#components.

52. *Routing* [online] [cit. 2021-02-14]. Dostupné z: https://laravel.com/docs/8.x/routing.

53. *Single Action Controllers* [online] [cit. 2021-02-15]. Dostupné z: https://laravel.com/docs/8.x/controllers#single-action-controller.

54. *Resource Controllers* [online] [cit. 2021-02-15]. Dostupné z: https://laravel.com/docs/8.x/controllers#resource-controllers.

55. *Twig* [online]. [B.r.] [cit. 2021-02-16]. Dostupné z: https://twig.symfony.com/.

56. *Blade Templates* [online] [cit. 2021-02-16]. Dostupné z: https://laravel.com/docs/8.x/blade.

57. *Amazon Web Services* [online] [cit. 2021-02-17]. Dostupné z: https://aws.amazon.com/.

58. *Microsoft Azure: Cloud Computing Services* [online] [cit. 2021-02-17]. Dostupné z: https://azure.microsoft.com/en-us/.

59. *DigitalOcean – The developer cloud* [online] [cit. 2021-02-17]. Dostupné z: https://www.digitalocean.com/.

60. POLOCK, Greg. *DigitalOcean vs AWS: Which Cloud Server is Better?* [Online] [cit. 2021-02-17]. Dostupné z: https://www.upguard.com/blog/digitalocean-vs-aws.

61. *Simple Storage Service (Amazon S3) - Amazon AWS* [online] [cit. 2021-02-17]. Dostupné z: https://aws.amazon.com/s3/.

62. SHATILIN, Ilja. *The dangers of public IPs* [online] [cit. 2021-02-18]. Dostupné z: https://www.kaspersky.com/blog/public-ip-dangers/24745/.

63. CAMERON, Lori. *How Netflix Adopted the New Mindset of Software Failure as the Rule—Not the Exception—and Survived the Great 2015 Amazon Web Services Outage* [online] [cit. 2021-02-18]. Dostupné z: https://www.computer.org/publications/tech-news/research/realizing-software-reliability-in-the-face-of-infrastructure-instability.

64. *HTTP Vs HTTPS: An In-Depth Comparison Of Features And Performance* [online] [cit. 2021-02-18]. Dostupné z: https://www.softwaretestinghelp.com/http-vs-https/.

65. *GitHub Actions* [online]. [B.r.] [cit. 2021-02-20]. Dostupné z: https://github.com/features/actions.

66. WU, Bo-Yi. *SSH for GitHub Actions* [online]. [B.r.] [cit. 2021-02-20]. Dostupné z: https://github.com/appleboy/ssh-action.

67. *Laravel Envoy* [online]. [B.r.] [cit. 2021-02-20]. Dostupné z: https://laravel.com/docs/8.x/envoy.

68. *SQL Injection* [online] [cit. 2020-12-11]. Dostupné z: https://owasp.org/www-community/attacks/SQL_Injection.

69. CORPORATION, Oracle. *MySQL Předpřipravené dotazy* [online] [cit. 2020-12-11]. Dostupné z: https://dev.mysql.com/doc/refman/8.0/en/sql-prepared-statements.html.

70. *Cross-site scripting* [online] [cit. 2020-12-11]. Dostupné z: https://owasp.org/www-community/attacks/xss/.

71. BADGER, ERIC NII SOWAH. *Phishing in depth* [online] [cit. 2020-12-11]. Dostupné z: https://owasp.org/www-chapter-ghana/assets/slides/OWASP_Presentation_FINAL.pdf.

72. PEJŠA, Jan. *Co je Cross-Site Request Forgery a jak se mu bránit* [online] [cit. 2020-12-28]. Dostupné z: https://www.zdrojak.cz/clanky/co-je-cross-site-request-forgery-a-jak-se-branit/.

73. *Brute Force Attack: Definition and Examples* [online] [cit. 2020-12-28]. Dostupné z: https://www.kaspersky.com/resource-center/definitions/brute-force-attack.

74. *What is a DDoS Botnet?* [Online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/.

75. LARSNOODEN. *SSH/OpenSSH/Keys* [online] [cit. 2021-01-27]. Dostupné z: https://help.ubuntu.com/community/SSH/OpenSSH/Keys.

76. *What is a Denial-of-Service (DoS) Attack?* [Online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/glossary/denial-of-service/.

77. *What is a DDoS Attack?* [Online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/.

78. *Slowloris DDoS Attack* [online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/.

79. *Tuning NGINX for Performance* [online] [cit. 2021-01-27]. Dostupné z: https://www.nginx.com/blog/tuning-nginx.

80. *UDP Flood Attack* [online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/.

81. *SYN Flood Attack* [online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/.

82. *HTTP Flood Attack* [online] [cit. 2021-01-27]. Dostupné z: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/.

83. *DDoS Attacks* [online] [cit. 2021-01-27]. Dostupné z: https://www.imperva.com/learn/ddos/ddos-attacks/.

84. *DNS Flood* [online] [cit. 2021-01-27]. Dostupné z: https://www.imperva.com/learn/ddos/dns-flood/.

Seznam obrázků

Slovník

ACID Atomicity, consistency, isolation, durability. 11

AWS Amazon Web Services. 21

Bug Chyby programu zapříčiněné chybou, či nepozorností programátora. 14

CGI Common Gateway Interface. Rozhraní umožňující webovým serverům spouštět terminálové aplikace. 13

CI/CD Continuous integration and continuous delivery. Mechanismus pro automatické testování a nasazování softwaru. 27

CSRF Cross-site request forgery. 31

CSS Cascading Stylesheets. 20

DDoS Distributed Denial of Service. 33

Debugger Programy určené pro ladění programů. 16

DKIM DomainKeys Identified Mail. Systém certifikátů pro ověření autenticity odesílatele emailu. 21

DNS Domain Name System. 27, 34

DoS Denial of Service. 33

FD File Descriptor. 33

framework Předpřipravený základ pro tvorbu aplikací. 13, 14, 19–24

GUI Grafické uživatelské rozhraní. 16

HTML Hyper Text Markup Language. 18, 23, 31

HTTP Hyper Text Transport Protocol. 12, 13, 15, 22, 23, 26, 28, 31

HTTP Polling Periodické stahování dat za pomoci separátních HTTP požadavků.. 14

HW Hardware. 10, 26, 33

IDE Integrovaná vývojová prostředí. 16, 20

IP Internet Protocol. 27, 31, 33, 34

IP Hash Algoritmus využívající klientskou IP adresu pro vybrání backendového serveru. 10

Least Connections Algoritmus vybírající server podle nejnižšího počtu právě zpracovávaných požadavků. 10

MVP Minimum Viable Product. Nejmenší možná verze produktu, která je schopná plnit zadaný účel. 19

MŠMT Ministerstvo školství, mládeže a tělovýchovy České republiky. 19

NTP Network Time Protocol. 34

on-demand Ve chvili potřeby. Typ škálování, při kterém se daný zdroj automaticky přizpůsobuje objemu uživatelských požadavků. 10

open-source Software s otevřeným zdrojovým kódem. 19, 20

ORM Object-Relational Mapper. Software sloužící pro konverzi dat mezi relačním a objektovým modelem. 14

Perl Rodina dvou vyšších dynamických programovacích jazyků Perl 5 a Perl 6. 13

PHP Rekurzivní název programovacího jazyka PHP: Hypertext Preprocessor. 13–15, 21–23, 27, 28

pub-sub Publish-subscribe. Způsob předávání zpráv. Zprávy jsou organizovány do kanálů. Klienti se do těchto kanálů mohou zapojit a dostávat příslušné zprávy. 15

RDBMS Relational Database Management System. 11, 12, 19, 20, 28, 49

real-time V reálném čase. Zobrazování, či provádění akcí, v současnou chvíli nebo s minimální odezvou. 12, 14, 19

Round Robin Algoritmus sekvenčně rotující backendové servery ze skupiny dostupných serveru, tj. jeden po druhém. 10

scaffolding Předpřipravená souborová struktura pro vývoj projektu. 20

SES Simple Email Service. 21

směrovač Software rozhodující o tom, jaký kód bude spuštěn v odpovědi na uživatelský požadavek. 14

SQL Structured Query Language. 11, 12, 29, 30, 49

SSE Server-sent events. 14

SSH Secure Shell. 27, 29, 32

SSL Secure Sockets Layer. 12, 27

STDERR Standard Error. 27

STDOUT Standard Output. 27

SW Software. 10, 21, 26, 33

Trigger SQL kód spuštěný v návaznosti ná různé události. 11

UDP User Datagram Protocol. 34

Uložená procedura Pojmenovaný SQL kód uložený v RDBMS, který můžeme opakovaně spouštět. 11

webserver Webový server. 12, 13, 28

WS Web Sockets. 12, 15, 28

WYSIWYG What You See Is What You Get. 23

XSS Cross-site scripting. 30

šablonovací knihovna Knihovna ulehčující generování HTML. 14