



**Vyšší odborná škola  
a Střední průmyslová škola elektrotechnická  
Plzeň, Koterovská 85**

## **DLOUHODOBÁ MATURITNÍ PRÁCE S OBHAJOBOU**

**Téma:**

**Rezervační systém pro zápůjčku školní techniky**

**Autoři práce:** František Kasík, Lukáš Janda, Pavel Hudec

**Třída:** 4. H

**Vedoucí práce:** Antonín Neumann

**Dne:**

**Hodnocení:**



Vyšší odborná škola  
a Střední průmyslová škola elektrotechnická  
Plzeň, Koterovská 85

## Zadání dlouhodobé maturitní práce

Žák: František Kasík, Lukáš Janda, Pavel Hudec  
Třída: 4.H  
Studijní obor: 18-20-M/01 Informační technologie  
Zaměření: Vývoj aplikací  
Školní rok: 2024-2025

Téma práce: ***Rezervační systém pro zápůjčku školní techniky***

### ***Pokyny k obsahu a rozsahu práce:***

1. Navrhněte a vytvořte responzivní webovou aplikaci, která bude fungovat jako rezervační systém pro zápůjčku školní techniky.
2. Systém rozdělte na 2 části – uživatelská a administrátorská.
  - Uživatel bude mít možnost si vybrat techniku ze seznamu vybavení a následně zarezervovat.
  - Administrátor má přístup k přehledu všech nových žádostí o zápůjčky a má možnost každou žádost schválit nebo odmítnout. Dále má administrátor možnost přidávat a odebírat nové vybavení.
3. Navrhněte a vytvořte grafický návrh a wireframe uživatelského a administrátorského rozhraní.
4. Navrhněte a vytvořte schéma databáze.
5. Celý projekt bude realizován na linux serveru. Frontend bude vytvořen pomocí react a tailwind, backend pomocí frameworku laravel.
6. Aplikace bude obsahovat:
  - Přihlašovací formulář
  - Žák:
    - Katalog dostupné techniky
    - Stránka s detailem techniky a formulářem pro vytvoření žádosti o rezervaci
    - Přehled probíhajících a nadcházejících rezervací
    - Historie proběhlých rezervací
  - Administrátor:
    - Přehled probíhajících a nadcházejících rezervací
    - Seznam žádostí o vypůjčení

- Historie všech proběhlých rezervací
- 7. Aplikace bude fungovat na veřejně dostupném serveru.
- 8. Zabezpečte server a všechny služby.
- 9. Zrealizujte monitoring a zálohování serveru/služeb.

**Určení částí tématu zpracovávaných jednotlivými žáky:**

- **František Kasík:**
  - Navrhněte uživatelské rozhraní, wireframe a logo webové stránky.
  - Implementujte responzivní a přehledný frontend pro uživatele a administrátory.
  - Provedte dokumentaci.
- **Lukáš Janda:**
  - Navrhněte a vytvořte schéma databáze.
  - Vytvořte veškerý backend stránky.
  - Vytvořte a použijte pro stránku unit testy.
  - Provedte dokumentaci.
- **Pavel Hudec:**
  - Nasaďte webovou aplikaci na server pomocí dockeru.
  - Provedte instalaci a správu webových služeb dle potřeb ostatních členů v týmu.
  - Provedte zabezpečení serveru a všech služeb + ověření (pentesty, aj.).
  - Monitoring serveru/služeb.
  - Zálohování serveru/služeb.
  - Provedte dokumentaci.

**Požadavek na počet vyhotovení maturitní práce:** 2 výtisky každý žák

Termín odevzdání: **11. dubna 2025**

Čas obhajoby: **45 minut (15 minut/žáka)**

Vedoucí práce: **Antonín Neumann**

Projednáno v **katedře** VTT a schváleno ředitelem školy.

V Plzni dne: 27. září 2024

Mgr. Jan Syřínek

Zástupce ŘŠ, zástupce statutárního orgánu

## **Anotace**

Maturitní práce se zaměřuje na tvorbu rezervačního systému pro zápůjčku školní techniky. Tato webová aplikace řeší různé problémy správy školního vybavení nabízeného jednotlivými učiteli školy žákům VOŠ a SPŠE Plzeň. Mezi tyto problémy doposud patřila neinformovanost žáků a nepřehledná a nekonzistentní správa výpůjček s absencí možnosti rezervace na určitý termín. Jedná se o responzivní a snadno použitelný systém vytvořený pomocí moderních technologií jako JavaScriptové knihovny React a PHP frameworku Laravel.

## **Klíčová slova**

Docker, Laravel, React, Rezervační systém, Školní vybavení, Web

„Prohlašujeme, že jsme tuto práci vypracovali samostatně a použili literárních pramenů a informací, které citujeme a uvádíme v seznamu použité literatury a zdrojů informací.“

„Souhlasíme s využitím naší práce učiteli VOŠ a SPŠE Plzeň k výuce.“

V Plzni dne: .....

Podpis: .....

# Obsah

|   |    |
|---|----|
| ÚVOD.....                               | 7  |
| 1 UŽIVATELSKÁ PŘÍRUČKA .....            | 8  |
| 1.1 ŽÁK .....                           | 8  |
| 1.1.1 Katalog .....                     | 8  |
| 1.1.2 Vytvoření rezervace.....          | 8  |
| 1.1.3 Přehled rezervací .....           | 9  |
| 1.1.4 Notifikace .....                  | 9  |
| 1.2 SPRÁVCE (UČITEL).....               | 10 |
| 1.2.1 Struktura panelu.....             | 10 |
| 1.2.2 Schválení rezervace .....         | 10 |
| 1.2.3 Předání vybavení .....            | 11 |
| 1.2.4 Navrácení vybavení .....          | 11 |
| 1.2.5 Vytvoření a úprava vybavení.....  | 11 |
| 1.2.6 Správa kategorií.....             | 12 |
| 1.2.7 Nastavení .....                   | 12 |
| 2 ANALÝZA A NÁVRH SYSTÉMU .....         | 13 |
| 2.1 SPECIFIKACE POŽADAVKŮ .....         | 13 |
| 2.1.1 Funkční požadavky – žák .....     | 13 |
| 2.1.2 Funkční požadavky – správce ..... | 14 |
| 2.2 POUŽITÉ TECHNOLOGIE .....           | 15 |
| 2.2.1 Frontend.....                     | 15 |
| 2.2.2 Backend .....                     | 16 |
| 2.2.3 Nasazení, monitoring a jiné.....  | 16 |
| 2.3 STRUKTURA DATABÁZE .....            | 17 |
| 2.3.1 ER diagram.....                   | 18 |
| 2.3.2 Popis tabulek a vztahů.....       | 18 |
| 2.4 NÁVRH A TVORBA ROZHRANÍ .....       | 19 |

|       |                                      |    |
|-------|--------------------------------------|----|
| 2.4.1 | Přehled rezervací žáka.....          | 20 |
| 2.4.2 | Katalog vybavení.....                | 20 |
| 2.4.3 | Detail vybavení.....                 | 21 |
| 2.4.4 | Administrátorský dashboard.....      | 21 |
| 2.4.5 | Detail rezervace.....                | 22 |
| 3     | FRONTEND.....                        | 23 |
| 3.1   | VZHLED A FUNKCIONALITA APLIKACE..... | 23 |
| 3.1.1 | Uživatelská část.....                | 23 |
| 3.1.2 | Administrátorská část.....           | 26 |
| 3.2   | NÁSTROJ INERTIA.JS.....              | 29 |
| 3.3   | ADRESÁŘOVÁ STRUKTURA.....            | 29 |
| 3.3.1 | Soubor app.css.....                  | 30 |
| 3.3.2 | Kořenový soubor.....                 | 30 |
| 3.3.3 | Komponenty.....                      | 31 |
| 3.3.4 | Layouty.....                         | 31 |
| 3.3.5 | Stránky.....                         | 31 |
| 3.4   | LAYOUTY.....                         | 32 |
| 3.4.1 | UserLayout a AdminLayout.....        | 32 |
| 3.4.2 | NoLayout a CenteredLayout.....       | 32 |
| 3.5   | PŘESMĚROVÁNÍ.....                    | 33 |
| 3.6   | PŘEDÁVÁNÍ DAT.....                   | 34 |
| 3.6.1 | Z backendu na frontend.....          | 34 |
| 3.6.2 | Z frontendu na backend.....          | 34 |
| 3.6.3 | Formuláře.....                       | 35 |
| 3.7   | VÝPIS DAT.....                       | 35 |
| 3.7.1 | Výpis prostých hodnot.....           | 35 |
| 3.7.2 | Výpis polí.....                      | 36 |
| 3.7.3 | Podmíněné stylování a výpis.....     | 36 |



|       |   |    |
|-------|---|----|
| 3.8   | KOMPONENTY .....                        | 36 |
| 3.8.1 | FlashMessages.jsx .....                 | 36 |
| 3.8.2 | ConfirmModal.jsx .....                  | 37 |
| 3.8.3 | BookingCalendar.jsx .....               | 37 |
| 3.8.4 | Pagination.jsx .....                    | 38 |
| 3.9   | MOŽNÁ VYLEPŠENÍ.....                    | 38 |
| 3.9.1 | Recenze .....                           | 39 |
| 3.9.2 | Správa uživatelů .....                  | 39 |
| 3.9.3 | Rezervace učitelů.....                  | 39 |
| 3.9.4 | Lepší časová omezení.....               | 39 |
| 3.9.5 | Neuvalovat postih.....                  | 39 |
| 4     | BACKEND.....                            | 40 |
| 4.1   | AUTENTIZACE.....                        | 40 |
| 4.1.1 | LDAP Integrace.....                     | 40 |
| 4.1.2 | Rate Limiting a ochrana přihlášení..... | 40 |
| 4.2   | AUTORIZACE .....                        | 41 |
| 4.2.1 | Role uživatelů (student/učitel).....    | 41 |
| 4.2.2 | Policies a Gates .....                  | 41 |
| 4.2.3 | Middleware.....                         | 42 |
| 4.3   | ZABEZPEČENÍ .....                       | 42 |
| 4.3.1 | Validace vstupů .....                   | 42 |
| 4.3.2 | CSRF ochrana.....                       | 43 |
| 4.3.3 | Transakce a konzistence dat .....       | 43 |
| 4.4   | SPRÁVA DAT .....                        | 44 |
| 4.4.1 | Struktura modelů a vztahů.....          | 44 |
| 4.4.2 | Soft Delete a archivace dat.....        | 44 |
| 4.4.3 | Filtrace a vyhledávání.....             | 45 |

|       |  |    |
|-------|--|----|
| 4.5   | REZERVAČNÍ LOGIKA .....                    | 46 |
| 4.5.1 | Životní cyklus rezervace.....              | 46 |
| 4.5.2 | Kontrola dostupnosti vybavení.....         | 46 |
| 4.5.3 | Zpracování kolizí rezervací .....          | 46 |
| 4.5.4 | Automatizované akce .....                  | 47 |
| 4.6   | NOTIFIKAČNÍ SYSTÉM .....                   | 47 |
| 4.6.1 | Emailové notifikace.....                   | 47 |
| 4.6.2 | Flash messages .....                       | 48 |
| 4.7   | OPTIMALIZACE VÝKONU .....                  | 48 |
| 4.7.1 | Zpracování obrázků .....                   | 48 |
| 4.7.2 | Queue Jobs .....                           | 48 |
| 4.7.3 | Cachování a lazy loading.....              | 49 |
| 4.8   | INERTIA.JS INTEGRACE .....                 | 49 |
| 4.8.1 | Komunikace s frontendem.....               | 49 |
| 4.8.2 | Předávání dat .....                        | 50 |
| 4.9   | TESTOVÁNÍ .....                            | 50 |
| 4.9.1 | Unit testy .....                           | 50 |
| 4.9.2 | Feature testy .....                        | 50 |
| 5     | NASAZENÍ A ZABEZPEČENÍ .....               | 52 |
| 5.1   | DEPLOYMENT NA LINUX SERVER.....            | 52 |
| 5.2   | ZABEZPEČENÍ APLIKACE .....                 | 52 |
| 5.2.1 | Zabezpečená komunikace pomocí HTTPS.....   | 52 |
| 5.2.2 | Implementace Content Security Policy ..... | 53 |
| 5.2.3 | Firewall.....                              | 53 |
| 5.2.4 | Autentizace a autorizace.....              | 53 |
| 5.2.5 | Bezpečnostní audit Docker kontejnerů.....  | 54 |
| 5.2.6 | Konfigurace produkčního prostředí.....     | 54 |
| 5.2.7 | Zálohování a redundance dat.....           | 54 |

|       |   |    |
|-------|---|----|
| 5.2.8 | Shrnutí bezpečnostních opatření.....      | 54 |
| 5.3   | MONITORING .....                          | 55 |
| 5.3.1 | Architektura monitorovacího systému.....  | 55 |
| 5.3.2 | Popis komponent .....                     | 55 |
| 5.3.3 | Implementace monitorovacího řešení.....   | 56 |
| 5.3.4 | Nastavení exportérů pro sběr metrik ..... | 57 |
| 5.3.5 | Konfigurace Grafany .....                 | 58 |
| 5.3.6 | Konfigurace Kuma .....                    | 59 |
| 5.3.7 | Monitorované endpointy .....              | 59 |
| 5.4   | ZÁLOHOVÁNÍ.....                           | 59 |
| 5.4.1 | Zálohování systému.....                   | 59 |
| 5.4.2 | Strategie zálohování .....                | 60 |
| 5.4.3 | Implementace zálohovacího systému .....   | 60 |
| 5.4.4 | Obsah záloh .....                         | 60 |
| 5.4.5 | Plánování záloh .....                     | 60 |
| 5.4.6 | Úložiště záloh .....                      | 61 |
| 5.4.7 | Rotace a správa záloh .....               | 61 |
| 5.4.8 | Bezpečnostní aspekty .....                | 61 |
| 5.4.9 | Monitorování zálohovacího procesu .....   | 61 |
| 6     | CO SE NEPOVEDLO .....                     | 63 |
| 6.1   | NÁVRH ŘEŠENÍ .....                        | 63 |
|       | ZÁVĚR.....                                | 66 |
|       | LITERATURA .....                          | 67 |
|       | SEZNAM OBRÁZKŮ .....                      | 69 |

## Úvod

VOŠ a SPŠE Plzeň disponuje rozsáhlým inventářem technického vybavení, které je k dispozici žákům pro jejich vzdělávací a projektové aktivity. Dosavadní způsob správy výpůjček však postrádal systematický přístup a přehlednost, což vedlo k neefektivnímu využívání dostupných zdrojů.

Tato skutečnost nás vedla k vytvoření webové aplikace, která řeší problematiku rezervačního systému školního vybavení. Hlavním cílem práce bylo vytvořit intuitivní platformu, která žákům umožní snadný přístup k přehledu dostupného vybavení a jeho rezervaci, zatímco vyučujícím poskytne efektivní nástroje pro správu výpůjček a monitoring využití zařízení. Systém byl navržen s ohledem na několik klíčových aspektů: integrace se školním LDAP systémem pro bezpečné přihlašování skrz školní účet, systém správy dostupnosti vybavení, přehledná správa rezervací s možností sledování historie výpůjček a responzivní design pro snadný přístup z různých zařízení.

Při vývoji byly využity moderní technologie a postupy, včetně frameworku Laravel pro backend, Reactu pro frontend a Tailwind CSS pro stylování. Pro zajištění maximální přenositelnosti a standardizace vývojového prostředí byl celý projekt kontejnerizován pomocí Dockeru, což významně usnadňuje nejen nasazení a správu aplikace, ale i její budoucí vývoj.

Práce je strukturována do několika hlavních částí. Tou první je uživatelská příručka. Po ní následuje část věnující se analýze požadavků a návrhu systému. Zbylé části tvoří technický popis implementace rozdělený na frontend a backend. Závěrečná část se zabývá nasazením systému na produkční server.

# 1 Uživatelská příručka

Tato část dokumentace slouží jako návod pro správné používání aplikace. Je určena pro žáky, i pro případné správce (učitele).

Pro přihlášení do systému je nutné zadat stejné přihlašovací údaje, které se využívají napříč celou školní sítí. Z bezpečnostních důvodů máte celkem 5 pokusů na přihlášení, po kterých následuje zablokování po dobu 1 minuty. Po úspěšném přihlášení budete přesměrováni na patřičnou stránku.

## 1.1 Žák

Pro navigaci slouží horní lišta. Pro odhlášení z aplikace klikněte v navigaci na své jméno pro otevření rozbalovacího menu.

### 1.1.1 Katalog

Pro zobrazení rezervovatelného vybavení využijte odkazu „Katalog“ v horní navigaci. U každého vybavení se zobrazují základní informace a dostupnost na následující týden. Pro nalezení specifického vybavení si lze výpis upravovat pomocí filtrů v horní části. Vyhledávání mezi položkami probíhá nejen na základě názvu ale i popisku.

Kliknutím na konkrétní vybavení budete přesměrováni na stránku s více informacemi, která ve spodní části zobrazuje přehled aktuálních a budoucích rezervací ostatních žáků. Této informace můžete využít např. pro získání zpětné vazby či rad, jak s vybavením zacházet. Zároveň je nutné mít na paměti, že i vaše žádost s vaší poznámkou zde bude veřejně viditelná všem ostatním žákům.

### 1.1.2 Vytvoření rezervace

Pro vytvoření rezervace přejděte na detail vybraného vybavení a klikněte na tlačítko "rezervovat". Tím dojde k otevření rezervačního kalendáře.

Nezabarvené dny představují data, kdy vybavení není nikým rezervované. Je-li nabízeno více jak 1 kus vybavení, je možné rezervaci překrývat s jinými studenty, a to do té doby, dokud není den zbarven červeně.

Pro každou rezervaci platí určitá pravidla, která jsou však automaticky kontrolována a není nutné se obávat, že by došlo k chybě. Přesto je dobré vědět, proč některé dny neumožňují konec či začátek vaší rezervace.

Základním pravidlem je, že žádná rezervace nemůže začínat ani končit o víkendu, protože vybavení musí být vyzvednuto a vráceno ve škole správci daného vybavení, což je o víkendu nemožné. Dále nesmí překrývat již plně rezervované (červené) dny. Zároveň nemůže začínat ve stejný den, kdy byla vytvořena a ani den po, aby měl správce dostatek času na posouzení a schválení rezervace. Každý správce může dokonce zablokovat určité dny např. z důvodu absence ve škole.

Den začátku rezervace je dnem vyzvednutí a den konce dnem vrácení. To v praxi znamená, že vybavení rezervované na 1 den musí být ve stejný den vyzvednuto i vráceno.

Po vybrání data rezervace je nutné před odesláním vyplnit důvod rezervace. Ten by měl být stručný a informovat nejen správce, ale i ostatní žáky o tom, jak hodláte vybavení využívat. Po odeslání žádosti obdržíte na školní email notifikaci o úspěšném vytvoření.

Rezervaci můžete zrušit pouze ve stavu „neschváleno“ či „schváleno“. Po dřívější dohodě se správcem je možné vypůjčené vybavení předčasně vrátit.

### **1.1.3 Přehled rezervací**

Každá rezervace má svůj životní cyklus. První fází představuje výše zmíněná žádost, kterou musí schválit správce. Lze ji kdykoliv zrušit a při schvalování se vedle řady dalších faktorů bere ohled na vaše proběhlé rezervace.

Do další fáze žádost přechází po jejím schválení. V této podobě je do vyzvednutí rezervovaného kusu vybavení. Je nutné si pro vybavení přijít až v den zahájení. Pozdní vyzvednutí je v systému zaznamenáno a použito při schvalovacím procesu budoucích žádostí. Při vyzvednutí je zaznamenán původní technický stav. Schválenou rezervaci je taktéž možné zrušit. Pokud však bude zrušena v době, kdy vaše rezervace měla již probíhat, bude do systému zaznamenána jako nevyzvednutá.

Probíhající rezervace nelze pochopitelně zrušit a je nutné ji včas vrátit. Před uložením do archivu dochází ke kontrole a zaznamenání konečného stavu.

### **1.1.4 Notifikace**

Veškeré notifikace jsou odesílány na vaši školní emailovou adresu. Budete notifikováni ohledně úspěšného odeslání a následného schválení rezervace. Taktéž den před vyzvednutím vybavení a v den jeho vrácení. Správce vybavení si vyhrazuje právo vaši rezervaci zamítnout, o čemž budete taktéž informováni.

## 1.2 Správce (učitel)

Pro navigaci slouží levý sloupec. Administrátorské rozhraní lze pohodlně používat jak na počítači, tak i mobilním telefonu.

### 1.2.1 Struktura panelu

Stránka „Dashboard“ nabízí v horní části rychlý přehled o aktuálním počtu různých druhů rezervací. Na stejné stránce je k dispozici jednoduchá statistika v podobě spojnicového grafu. Ten zobrazuje počty nových rezervací připadajících na jednotlivé dny. Rozsah grafu lze změnit na týden, měsíc či rok. Další důležité informace poskytuje žebříček top 3 vašich vybavení. Této informace lze využít při pořizování nebo údržbě vybavení a je možné se pomocí tlačítka odkázat na kompletní seznam.

Další stránkou v pořadí je „Kalendář“, který nabízí přehledné zobrazení všech druhů rezervací. Při kliknutí na libovolnou rezervaci budete přesměrováni na stránku pro správu specifické rezervace obsahující bližší detaily a proveditelné akce.

Vedle kalendáře lze rezervace a žádosti spravovat pomocí specifických výpisů. U každého druhu se zobrazuje počet, aby nebylo nutné se pro přehled vracet na „Dashboard“.

Další stránky slouží pro správu a přidávání vybavení a kategorií. Stránka akce plní pouze informační roli o automatických událostech, které se pravidelně provádějí na pozadí. Součástí panelu je také manuál.

Pro zpřístupnění stránky s nastavením účtu klikněte na své jméno ve spodní části levého navigačního panelu.

### 1.2.2 Schválení rezervace

Při vytvoření žádosti žákem obdržíte notifikaci na školní email vyzívající k posouzení a případnému schválení či odmítnutí. Žádost je viditelná buď v kalendáři akcí a je označena červenou barvou, nebo na stránce „Neschválené“.

Po kliknutí na rezervaci v kalendáři či na ikonu oka ve výpisu budete přesměrováni na posuzovací stránku. Zde vám budou k dispozici informace o žádaném vybavení, délce, začátku a konci rezervace. Taktéž je k dispozici historie rezervací žáka nejen vámi spravovaného vybavení, ale i ostatních učitelů.

Není nutné se obávat, že by se nějaká žádost kryla s jinou, jelikož to kontroluje systém. Pokud nedojde ke schválení žádosti před jejím zahájením, tak dojde po uplynutí určitého počtu dnů k jejímu automatickému zrušení. Při potvrzení můžete zadat krátkou poznámku k rezervaci,

která bude žákovi dostupná. Lze ji změnit či přidat i v následujících krocích. Při zamítnutí bude obsah této poznámky použit jako důvod zamítnutí v notifikačním emailu pro žáka. Zamítnuté žádosti nejsou v systému nijak zaznamenávány.

### **1.2.3 Předání vybavení**

Po schválení by se měl žák v den začátku rezervace dostavit na místo uvedené u vybavení. Je tedy vhodné již při schvalování do poznámky uvést čas, kdy budete schopni vybavení předat.

Při předání je nutné v systému schválenou rezervaci zahájit. Formulář pro zahájení je téměř totožný s tím pro schválení. Jedinou změnu představuje možnost zadat technický stav při předání.

### **1.2.4 Navrácení vybavení**

Před zrušením se musíte přesvědčit, zda vybavení bylo skutečně vráceno, jelikož se nejedná o navratitelnou akci. Před ukončením je důležité nezapomenout na kontrolu a zadání koncového stavu. Po ukončení rezervace dojde k jejímu přesunutí do archivu, kde je rezervace po dobu 4 let před tím, než se automaticky odstraní.

U archivované rezervace jde vidět všechny důležité informace a případné problémy, které s rezervací nastaly v průběhu jejího životního cyklu.

### **1.2.5 Vytvoření a úprava vybavení**

Správa veškerého vybavení probíhá na stránce „Vybavení“. Pro vytvoření využijte tlačítka „Přidat vybavení“. Dojde k přesměrování na formulář, kde je nutné vyplnit veškerá pole až na náhledový obrázek a kategorii, které lze zpětně přidat.

Je možné nahrát náhledový obrázek ve formátu jpg, jpeg, png nebo webp a to až do velikosti 10 MB. Po nahrání obrázku dojde vždy k jeho automatické optimalizaci.

Při úpravě vybavení vypadá formulář obdobně. Editační stránka nabízí možnost vybavení dočasně odstranit. Vybavení se od dané chvíle přestane zobrazovat v katalogu a nepůjde na něj vytvořit rezervaci. Zároveň dojde k odstranění všech neschválených i schválených rezervací pro tuto položku.

Této akce je vhodné využít např. při nenavrácení vypůjčené techniky včas či při jeho opravě. Probíhající rezervace zrušeny nebudou. Pokud chcete vybavení trvale odstranit, klikněte na tlačítko „Odstranit“, které se zobrazí po dočasném odstranění.

Při změně počtu kusů např. při rozbití dochází ke kontrole toho, zda nenastává překrývání rezervací. V případě, kdy jsou nabízeny 2 kusy vybavení k vypůjčení a dochází k tomu, že jsou



či budou ve stejný moment vypůjčeny oba a vy se rozhodnete 1 vybavení odstranit, dojde k zamítnutí akce. Buď ručně zrušíte jednu z rezervací, počkáte, než k překrývání nebude docházet, nebo ho dočasně odstraníte.

### **1.2.6 Správa kategorií**

Všechny kategorie jsou dostupné na stránce „Kategorie“. Na rozdíl od inventáře vybavení se jedná o sdílenou část mezi všemi správci. To znamená že zde můžete vidět výpis všech kategorií i od jiných učitelů. To je z důvodu, aby se zabránilo k jejím opakováním.

Při úpravě názvu jakékoliv kategorie dojde k zaznamenání jména editora i času, kdy ke změně došlo. Je dokonce možné mazat kategorie jiných uživatelů. Lze však pouze smazat ty, na které není vázáno žádné vybavení. Při zobrazení detailu kategorie je dostupný výpis vašeho vybavení, které se na kategorii váže.

Pro vytvoření nové kategorie využijte tlačítka „Přidat kategorii“.

### **1.2.7 Nastavení**

Na stránce nastavení se nachází možnost uvést defaultní místnost pro výdej vybavení. Tato hodnota se používá při vytváření nového vybavení a automaticky se dosazuje do pole „Místnost“. Její defaultní hodnotou je číslo vašeho kabinetu.

Dny nedostupnosti se udávají ve formátu čísel den v týdnu v rozmezí 1 (pondělí) – 5 (pátek), která jsou oddělena čárkami. Údaj 1,2 např. znamená, že žádná z rezervací na vámi spravované vybavení nemůže začínat ani končit v pondělí a v úterý.

## 2 Analýza a návrh systému

V rámci analýzy a návrhu systému jsme nejprve prozkoumali aktuální způsob správy školního vybavení určeného pro vypůjčení. Dosavadní systém byl charakterizován jako neefektivní a nedostatečně přehledný.

Žáci mnohdy nevěděli, co a kde je možné si vypůjčit. Učitelé si museli vést vlastní seznamy pro udržení přehledu o probíhajících výpůjčkách, což představovalo nekonzistentní řešení s nízkou flexibilitou, kde mohla nastat řada problémů.

Mezi základní požadavky patřila především potřeba efektivní správy školního vybavení, která umožní studentům jednoduchý přístup k přehledu dostupných zařízení a jejich rezervaci, a zároveň poskytne správcům nástroje pro správu výpůjček a monitoring využití techniky.

Dalšími klíčovými vlastnostmi měla být automatizace akcí, u kterých by byl systém závislý na přesnosti jak učitele, který vybavení spravuje, tak i žáka. Příkladem takové akce je kontrola, zda se jednotlivé rezervace nekryjí, čímž se zabrání situaci, kdy žák, přestože má vybavení rezervované, ho nemá v daný den k dispozici.

### 2.1 Specifikace požadavků

Tato část stanovuje základní předpoklady a očekávání, která měl rezervační systém splňovat. Jejím cílem je definovat, co systém musí umět z pohledu uživatelů i správy, a zároveň vymezit technologická omezení a podmínky, které ovlivňují následný návrh a implementaci.

#### 2.1.1 Funkční požadavky – žák

Při provedení libovolné akce by měl být uživatel (žák i správce) informován prostřednictvím krátké zprávy (flash message) o následcích či úspěšnosti provedené akce. Tímto je vedle notifikací pomocí emailu žáků zajištěno, že si je uživatel vždy jist tím, že daná akce byla úspěšně zpracována.

Katalog vybavení by měl žákovi se zájmem o vypůjčení vybavení přehlednou formou prezentovat seznam dostupného vybavení od různých učitelů. Na první pohled bez potřeby rozkliknout vybavení by měl mít žák k dispozici většinu potřebných údajů, mezi které patří: dostupnost pro stávající týden, fotografie vybavení, krátký popis, kategorie a výrobce. Zároveň by měla existovat možnost si celý výpis filtrovat na základě kategorie, řadit podle pořadí a vyhledávat pomocí nejen názvu, ale i popisu a parametrů.

Po rozkliknutí vybavení by měl být žák přesměrován na stránku s více detaily. Rozhodně by zde neměl chybět kompletní popis, jméno správce či místo vyzvednutí. Vedle toho by se na této stránce měl nacházet jednoduchý rezervační kalendář.

Žák by měl mít přehled o rezervovatelných či plně obsazených dnech. Kalendář by neměl dovolit, aby rezervace končila či začínala o víkendu. Zároveň nesmí být možné zvolit, aby rezervace začínala ve stejný den jako byla vytvořena, aby měl správce čas rezervaci posoudit. Uživatel by měl mít možnost zvolit rozsah dnů či samostatný den.

Žák má zároveň povinnost na žádost zadavatele práce napsat účel rezervace a jiné užitečné informace pro ostatní studenty – tato informace by měla být veřejně dostupná pro ostatní uživatele a zároveň by měla být součástí schvalovacího procesu žádosti.

Žák by měl mít možnost vidět aktuální stav a počet svých rezervací. V historii by vedle seznamu měl mít přístup i k informacím jako poznámka správce a problémy s rezervací. Měla by zde zároveň existovat možnost zrušení žádosti, aby se uvolnila kapacita pro vybavení.

### **2.1.2 Funkční požadavky – správce**

Správce by měl mít k dispozici jednoduchý a přehledný dashboard. Ten by mu měl vedle přehledu o počtu žádostí či právě probíhajících výpůjčkách zobrazovat základní statistiku, do které by měla být zahrnuta celková popularita každého vybavení, na základě které může snížit či naopak zvýšit jeho množství.

Pro lepší přehlednost by zde měla být možnost kalendářního zobrazení, které by mělo nabízet rychlý přehled o rezervacích a žádostech.

Měl by zároveň existovat i základní výpis žádostí a probíhajících rezervací do tabulek, které musí zobrazovat základní potřebné informace.

Dále musí existovat správa vybavení, kde správce bude mít přístup k přehlednému výpisu. Měla by existovat i možnost dočasného odstranění, které vybavení nesmaže, ale pouze znemožní jeho rezervování.

Je nutné mít k dispozici správu kategorií. Aby se předešlo duplicitám, tak by měla být tato část sdílena mezi všemi správci. Každá kategorie by měla jít změnit či smazat v případě, kdy není používána.

Jelikož je každý učitel považován za správce, je nutné mít k dispozici přehledný a stručný manuál i pro nezkušené uživatele. Vedle popisu fungování a práce s aplikací by měly být zmíněny i následky různých akcí.

## 2.2 Použité technologie

Při vývoji byla využita řada pro nás ze školy neznámých frameworků a knihoven, se kterými bylo nutné se postupně seznámit. Každá z použitých technologií disponuje rozsáhlou a funkční dokumentací, bez které bychom se neobešli.

Žádná technologie nebyla zvolena náhodou, ale skrze proces, během něhož nebyla posuzována pouze jednoduchost užití. Byl kladen důraz na kvalitní dokumentaci a širokou uživatelskou podporu znamenající jednodušší řešení případných problémů.

### 2.2.1 Frontend

Pro vývoj struktury uživatelského rozhraní byla využita JavaScriptová knihovna React. Ta byla zvolena díky rozsáhlému ekosystému knihoven a komunitě, vysoké poptávce na trhu práce a předešlým zkušenostem. Při vývoji bylo využito komponentového přístupu, který je pro React obvyklý, a znamená vysokou znovu použitelnost a snadnou údržbu. Dalším zásadním důvodem byla možnost tvorby tzv. jednostránkové aplikace (SPA – single page application). To znamená, že při přesměrování v rámci webové stránky nedochází k obnovení v prohlížeči.

Každá stránka byla nastylována pomocí CSS frameworku Tailwind. Velmi se podobá frameworku Bootstrap a pracuje se s ním podobným způsobem. Není nutné vymýšlet názvy tříd a následně přecházet do CSS souboru, ve kterém při tvorbě rozsáhlejší aplikace může docházet k opakování již definovaných stylů. Místo toho probíhá stylování přímo v HTML pomocí předdefinovaných tříd. Rozhodujícím rozdílem však byla celková velikost CSS. V případě Tailwind nejsou nepoužité třídy součástí finálního CSS souboru, což značně šetří zdroje a zaručuje rychlejší odezvu stránku. Zároveň je velmi jednoduché provést personalizovanou konfiguraci pomocí přiloženého konfiguračního souboru. Samozřejmě zde stále existuje možnost si napsat vlastní CSS třídy pro opakující či nepodporované styly.

Jelikož v případě použití Tailwind může docházet ke snížení přehlednosti HTML, je součástí aplikace i formátovač nazývaný Prettier. Ten je nakonfigurován tak, aby přidělené Tailwind třídy logickým způsobem seřadil. Primárně se však stará o automatické naformátování struktury HTML a JavaScriptu pro udržení přehlednosti a zjednodušení práce programátora.

Klíčovým vývojovým nástrojem je Vite. Ten se stará o efektivitu a rychlost aplikace při vývoji a následně o vytvoření finální optimalizované verze aplikace, která se nasazuje na server. Stará se o spouštění tzv. vývojového serveru, který při změně kódu frontendu automaticky obnoví stránku se zachováním původního stavu. Podporuje řadu knihoven a v současnosti se jedná o nezbytný nástroj pro moderní a efektivní vývojový proces webových stránek.

Nástroj Inertia.js působí jako most mezi frontendem a serverem. Dá se tedy zařadit jak mezi nástroje pro vývoj backendu, tak i frontendu. Díky tomu můžeme využívat jak výhody frameworků pro vytváření SPA aplikací, tak i výhody zpracovávání dat na serveru bez nutnosti vytváření REST API. Díky tomu se celá aplikace se vším všudy nachází v jednom společném adresáři. Data se předávají přímo Reactu, což zjednodušuje jejich synchronizaci a komunikaci. O routing včetně zabezpečení se tak může starat knihovna Laravel. Inertia.js zajišťuje celou řadu dalších věcí, toto je však výčet pouze těch nejzásadnějších.

Součástí aplikace pro vývoj frontendu je spousta menších podpůrných knihoven, které slouží k zjednodušení práce a zabezpečení aplikace. Obvykle se jedná o optimalizovaná a bezpečná řešení, která by nebylo s našimi dosavadními zkušenostmi možné replikovat.

### **2.2.2 Backend**

Pro vývoj backendu jsme zvolili framework Laravel. Byl vybrán díky jeho popularitě mezi vývojáři, rozsáhlé komunitě a přehledné dokumentaci.

V Laravelu je striktně stanovená adresářová struktura, což se může zdát limitující, avšak pro začátečníka to představuje ideální případ. Vždy je totiž jasné, co a kam se má umístit. Laravel používá architekturu MVC, která odděluje vzhled, logiku a data aplikace, díky čemuž je aplikace přehledná a udržitelná.

Hlavní výhodou je rozsáhlá komunita a ekosystém. Většina PHP knihoven má svou verzi přizpůsobenou pro Laravel, a tudíž se s nimi jednoduše pracuje. Příkladem je knihovna LdapRecord, kterou využíváme pro autentizaci uživatelů.

Je zde zabudována ochrana proti různým typům útoku jako XSS, CORS či SQL injection. S databázemi se manipuluje pomocí vestavěného nástroje Eloquent ORM. Díky němu jsme byli v průběhu vývoje schopni rychle implementovat změny databázové struktury a vytvářet vztahy mezi tabulkami. To nám umožnilo se více věnovat samotné funkcionalitě aplikace.

### **2.2.3 Nasazení, monitoring a jiné**

Pro nasazení aplikace jsme využili Docker, který umožňuje zapouzdřit celou aplikaci včetně všech závislostí do izolovaných kontejnerů. Díky tomu je možné zajistit konzistentní běhové prostředí bez ohledu na to, kde aplikace běží, což výrazně zjednodušuje přenos mezi vývojovým a produkčním prostředím. Docker compose následně slouží jako nástroj pro organizaci více kontejnerů. Definuje, jak spolu jednotlivé služby komunikují. V našem případě jsme vytvořili oddělené konfigurace pro vývojové a produkční prostředí, s důrazem na optimalizaci a bezpečnost v produkci.

Naši aplikaci jsme postavili na víceúrovňovém obrazu, který nejprve sestaví aplikaci v tzv. "builder" fázi a poté zkopíruje pouze potřebné soubory do finálního obrazu. Tímto přístupem jsme docílili výrazně menší velikosti výsledného kontejneru a zlepšili celkovou bezpečnost, jelikož produkční obraz neobsahuje vývojové nástroje a potenciální bezpečnostní rizika. Pro běh aplikace využíváme Nginx jako webový server, který zpracovává příchozí HTTP požadavky a směřuje je do PHP-FPM, kde běží samotná Laravel aplikace.

Pro monitoring aplikace jsme zřídili Prometheus jako databázi pro ukládání metrik a Grafanu pro jejich vizualizaci. Tento přístup nám umožňuje v reálném čase sledovat výkon aplikace, využití systémových zdrojů a rychlost odezvy. Pro sběr metrik z MySQL databáze a Nginx webového serveru jsme nasadili specializované exportéry, které poskytují detailní náhled do těchto komponent. Doplnuje je Kuma, jednoduchý nástroj pro monitorování dostupnosti, který průběžně kontroluje, zda je aplikace funkční a dostupná.

Pro zajištění bezpečnosti a integrity dat jsme vytvořili automatické zálohy s dvoutýdenním intervalem. Zálohy zahrnují jak kompletní databázi, tak zdrojový kód a konfigurační soubory, přičemž jsou automaticky vyloučeny dočasné soubory a závislosti, které lze znovu generovat. Vytvořené zálohy jsou primárně ukládány lokálně na serveru, kde je uchováván omezený počet nejnovějších verzí s automatickou rotací. Pro zvýšení spolehlivosti jsou zálohy následně automaticky nahrávány do cloudového úložiště MEGA, což poskytuje oddělené zabezpečení dat. Celý proces je řízen pomocí skriptů spouštěných plánovačem úloh cron a generuje podrobné logy pro snadnou kontrolu a diagnostiku.

Pro zjednodušení nasazení aktualizací jsme implementovali Supervisor, který spravuje a monitoruje procesy uvnitř kontejneru. Tento nástroj zajišťuje, že v případě pádu aplikace dojde k automatickému restartu, což výrazně zvyšuje spolehlivost a dostupnost celého systému. V produkčním prostředí navíc využíváme optimalizovanou konfiguraci PHP s důrazem na výkon a bezpečnost.

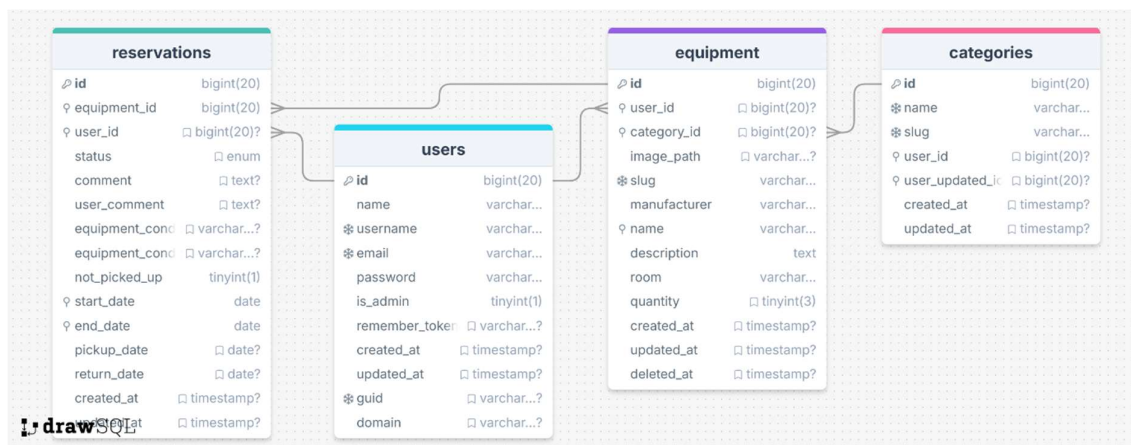
Celý tento ekosystém je navržen s ohledem na bezpečnost, škálovatelnost a jednoduchou údržbu, což odpovídá moderním přístupům k vývoji a nasazení webových aplikací.

## **2.3 Struktura databáze**

Databázový model rezervačního systému je navržen tak, aby efektivně podporoval všechny funkce aplikace. Klíčové entity, jejich atributy i vztahy mezi nimi byly promyšleny s ohledem na zajištění integrity dat, rychlost dotazů a škálovatelnost.

### 2.3.1 ER diagram

ER diagram modelu ilustruje hlavní entity v systému a vztahy mezi nimi.



Obrázek 1: ER diagram

Diagram zachycuje následující entity:

- **users** – evidence uživatelů systému
- **categories** – kategorie vybavení
- **equipment** – informace o vybavení
- **reservations** – informace o rezervacích

Diagram také jasně znázorňuje, jak jsou tabulky propojeny cizími klíči (equipment odkazuje na kategorii i uživatele, reservations propojují uživatele a vybavení atd.).

### 2.3.2 Popis tabulek a vztahů

V následující části je popsán význam a role jednotlivých tabulek.

Tabulka users obsahuje informace o uživateli (jméno, username, email, heslo atd.) a slouží jako centrální prvek pro autentizaci a autorizaci. Vztahuje se na ostatní entity pomocí cizích klíčů, zejména rezervace a vybavení, které jsou evidovány s ohledem na uživatele, kteří je zadali.

Tabulka categories umožňuje logické dělení vybavení do skupin, což usnadňuje filtrování a třídění položek v katalogu. Každá kategorie je identifikována unikátním názvem a slugem.

Tabulka equipment obsahuje informace o vybavení. Tabulka obsahuje údaje jako název, výrobce, popis, množství, umístění či cestu obrázku. Každá položka je propojena s vybranou kategorií a uživatelem, který ji zadal, což umožňuje sledování odpovědnosti a historie změn.

Tabulka reservations obsahuje informace o rezervacích. Eviduje informace o tom, který uživatel si rezervoval konkrétní vybavení na určité období. Dále obsahují stavy rezervací, komentáře a indikátor, zda bylo zařízení vyzvednuto. Tato tabulka má cizí klíče na tabulky users a equipment, čímž zajišťuje integritu referenčních dat.

Tabulky jobs a job\_batches jsou využívány systémem pro asynchronní zpracování úloh (queue). Umožňují efektivní odesílání notifikací, zpracovávání batch úloh a obecnou podporu automatizovaných procesů v rámci systému, což je důležité pro škálovatelnost a výkonnost aplikace.

## 2.4 Návrh a tvorba rozhraní

Při tvorbě rozhraní byl kladen důraz na responzivitu a přehlednost. Bylo důležité vytvořit takové rozhraní, které se chová standardně a předvídatvě. To v tomto kontextu např. znamená, že na mobilních zařízeních se navigace stránky otevře po rozkliknutí tlačítka v pravém horním rohu obrazovky.

Kompletní návrh webových stránek byl realizován v programu Figma, což je moderní a dnes rozšířený nástroj pro tvorbu uživatelských rozhraní primárně pro webové aplikace. Návrh neodpovídá současnému vzhledu aplikace. V průběhu vývoje plnil především podpůrnou roli pro plánování rozložení prvků na stránce.

Responzivita každé stránky byla ručně ověřena na různých webových prohlížečích (Chrome, Firefox) i operačních systémech (IOS, Android, Windows). Každá stránka prošla kontrolou nástrojem Lighthouse od Googlu, který vyhodnocuje různé aspekty. Jedná se však pouze o základní nástroj a bylo nutné se řídit dalšími pravidly, která se však těžko ověřují. Příkladem je používání sémantických znaků, na které byl kladen velký důraz, jelikož s nimi pracují nejen prohlížeče, ale i čtečky, na kterých jsou závislí zrakově postižení lidé.

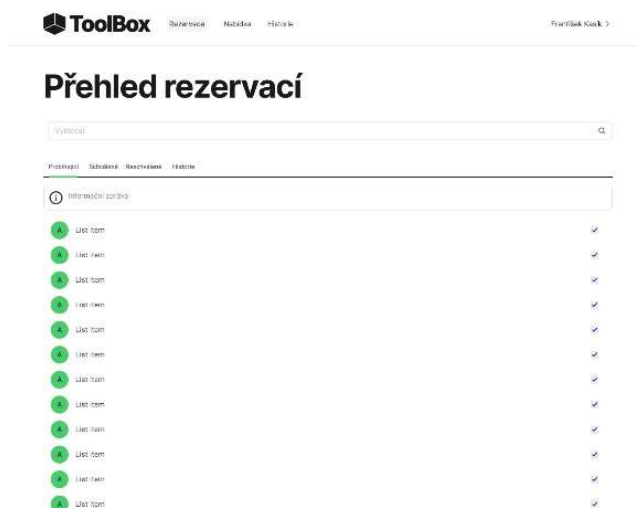
Finální podoba stránky využívá neutrální barevné schéma tvořené monochromatickými barvami. Akcentovou barvou představuje zelená v podobném odstínu jako logo školy.

Každá stránka byla vytvořena jak pro tmavý, tak i světlý režim. To znamená, že byly vytvořeny 2 podoby stránek s odlišnými paletami barev. Tento vizuál se automaticky přepíná na základě systémových preferencí uživatele. Představuje to lepší uživatelský zážitek, jelikož stránku je pohodlnější prohlížet v noci. O implementaci této funkcionality se stará CSS framework Tailwind.



## 2.4.1 Přehled rezervací žáka

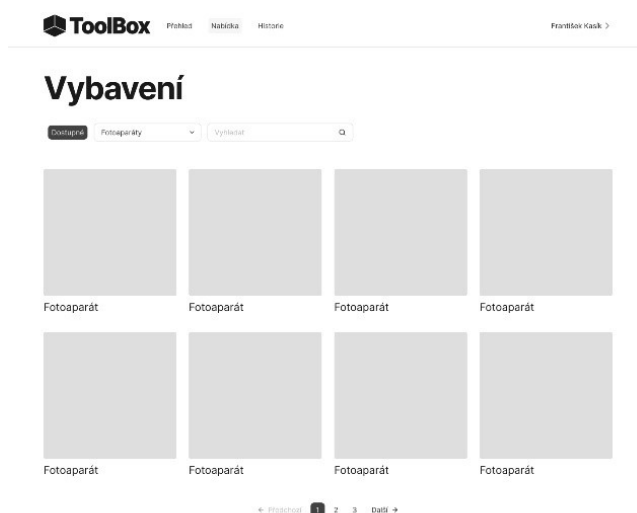
Jedná se o uživatelskou obrazovku žáka. Byla navržena s důrazem na přehlednost a jednoduchost použití. Výpis probíhá různými formami pro snadné odlišení mezi různými druhy rezervací. Vždy je přítomna miniatura náhledu vybavení pro jeho rychlou identifikaci.



Obrázek 2: návrh přehledu rezervací žáka

## 2.4.2 Katalog vybavení

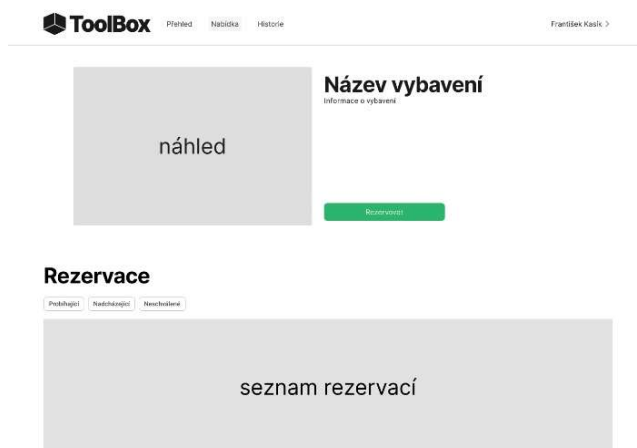
Katalog vybavení je klíčovou stránkou, která je navržena tak, aby přehlednou formou zobrazovala nezbytné informace. Každá položka vybavení se zobrazuje jako karta s názvem, obrázkem a základními informacemi vybavení.



Obrázek 3: návrh katalogu dostupného vybavení

### 2.4.3 Detail vybavení

Má poskytovat uživatelům podrobné informace o konkrétním zařízení včetně jeho popisu, dostupnosti a možnosti rezervace. Součástí stránky je přehled vázaných probíhajících, nadcházejících a neschválených rezervací.



Obrázek 4: návrh detailu vybavení z katalogu

### 2.4.4 Administrátorský dashboard

Dashboard je domovskou stránkou správce. Prezентuje základní informace o rezervacích a vybavení. Nejzajímavějším prvkem je graf zobrazující trendy v počtu rezervací a přehled nejčastěji rezervovaného vybavení.



Obrázek 5: návrh administrátorského dashboardu

### 2.4.5 Detail rezervace

Správci na této stránce mohou spravovat rezervace. Nepatrně se liší ve své funkčnosti na základě stavu rezervace. Vždy jsou k dispozici základní údaje a možné akce, které se připínají na konec stránky.



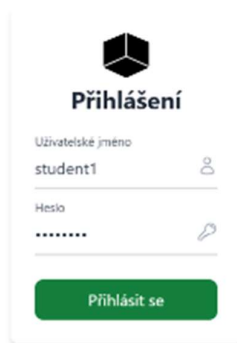
Obrázek 6: návrh schvalovací stránky rezervace

## 3 Frontend

### 3.1 Vzhled a funkcionálita aplikace

Rezervační systém se dělí celkem na 2 části – pro správce, kterého představuje každý učitel, a žáka. Oba přistupují k aplikaci přes jednotný přihlašovací formulář.

Jednotlivé části se od sebe liší nejen ve své funkcionalitě, ale i v rozložení. Jak bylo již zmíněno, každá stránka disponuje 2 motivy (tmavým a světlým), které se automaticky přepínají v závislosti na nastavení motivu operačního systému.

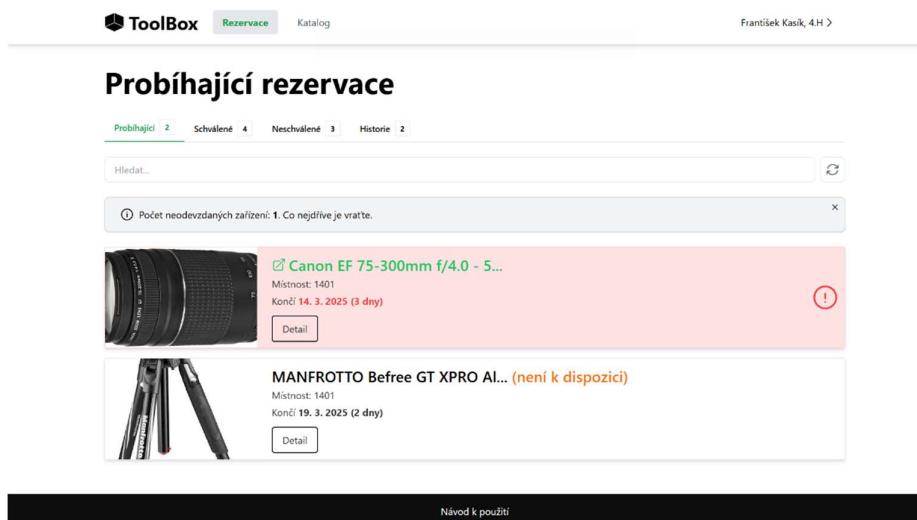
The image shows a mobile app login screen. At the top is a black cube icon. Below it is the title 'Přihlášení' in bold. There are two input fields: 'Uživatelské jméno' with the text 'student1' and a person icon, and 'Heslo' with a password mask '.....' and a key icon. At the bottom is a green button with the text 'Přihlásit se'.

Obrázek 7: přihlašovací formulář

#### 3.1.1 Uživatelská část

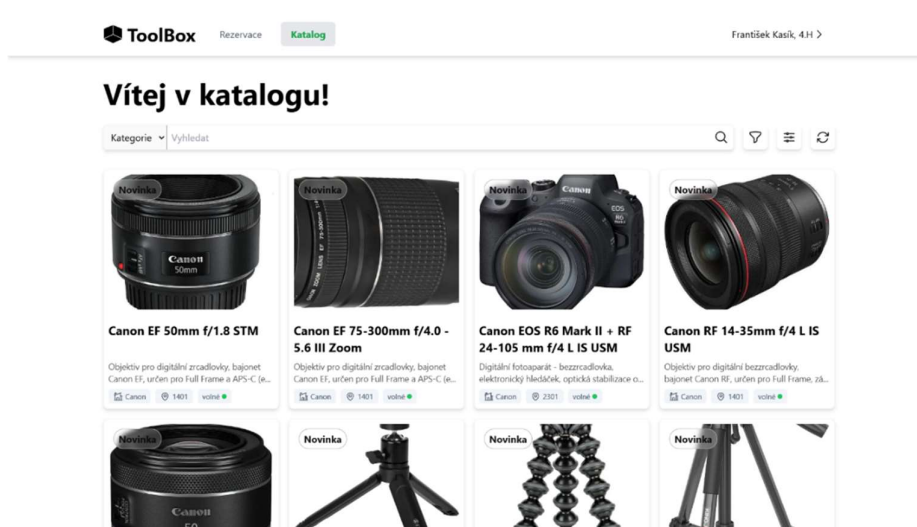
Běžný uživatel, kterého představuje každý žák, má přístup k několika stránkám. Těmi nejzásadnějšími je katalog a přehledy vlastních rezervací.

Přehled probíhajících rezervací představuje domovskou stránku, na kterou je žák přesměrován po přihlášení. To je z toho důvodu, že probíhající rezervace je nutné vracet včas pro vyhnutí se postihu. Pomocí lišty odkazů si může žák zobrazit jednotlivé typy rezervací od schválených až po archivované. Každá záložka je opatřena číslem zobrazujícím počet rezervací v určité kategorii, aby žák nemusel ručně procházet každou stránku. Výpis každého typu rezervace se od sebe stylisticky liší, aby žák mohl jednoduše identifikovat, který druh si aktuálně prohlíží. Výpisy doprovází informační zprávy a označování položek s nastalými problémy (pozdní odevzdání/vyzvednutí...).



Obrázek 8: přehled probíhajících rezervací

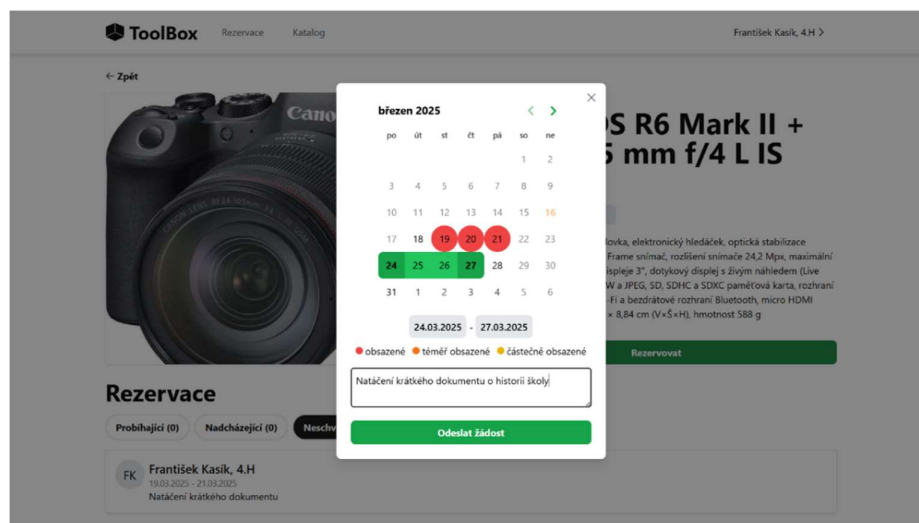
Nedílnou součástí systému tvoří katalog nabízející kompletní přehled všeho dostupného vybavení. Výpis je stránkovaný pro dosažení vyšší rychlosti načítání, jelikož je vždy nutné načíst menší množství dat. Zároveň to poskytuje uživateli lepší přehled. Při přesměrování na stránku katalogu dochází nejprve k načtení základní struktury stránky a až poté se načítají informace o produktech. Při čekání na načtení informací o produktech se zobrazuje tzv. skeleton, který se po načtení dat ze serveru zamění za vybavení. Této funkcionalitě se říká „deferred props“ a je součástí nové verze nástroje Inertia, na kterou jsme byli schopni úspěšně přejít v pozdní fázi vývoje.



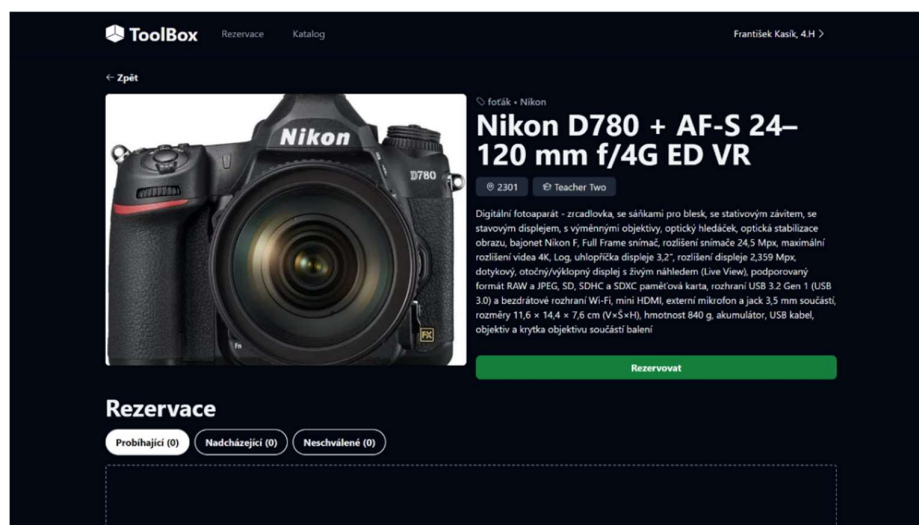
Obrázek 9: katalog vybavení

Pro zobrazení více informací a rezervaci slouží detailní stránka vybavení. Rezervační kalendář se zobrazuje jako modální (vyskakovací) okno a umožňuje žákovi si vytvořit rezervaci pouze

na specifické dny. Rezervace v kalendáři je možné překrývat pouze v případě, kdy určité vybavení disponuje více jak 1 kusem. Do kalendáře se projevují i preference správce v podobě dnů nedostupnosti. Kontrola správnosti rezervace probíhá nejen na straně prohlížeče, ale i na straně serveru.



Obrázek 10: detailní stránka s otevřeným kalendářem

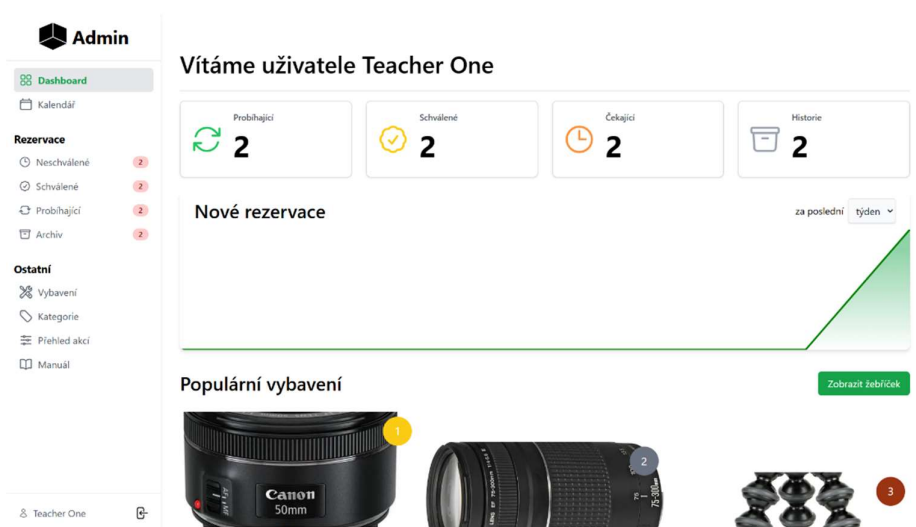


Obrázek 11: detailní stránka v tmavém režimu

### 3.1.2 Administrátorská část

Administrátorská část je určena pouze správcům, za které jsou považováni všichni učitelé školy.

Domovskou stránkou je „Dashboard“. Jedná se o přehledný panel dávající danému správci většinu potřebných informací na 1 místě. Prvně si může všimnout v horní části obrazovky přehledu počtu jednotlivých druhů rezervací. Na každou kategorii je možné kliknout, což způsobí přesměrování na stránku s výpisem vybraného typu rezervací. Další sekci tvoří spojnicový graf, který zobrazuje využití vybavení správce pro jednotlivé dny. Rozsah grafu si lze upravovat pomocí select boxu v horním pravém rohu na poslední týden, měsíc či rok. Poslední část tvoří žebříček populárního vybavení zobrazující 3 nejvíce vypůjčované. Pro zobrazení celého inventáře seřazeného podle popularity lze využít tlačítka v pravém horním rohu.



Obrázek 12: administrátorský dashboard


Stránka „Kalendář“ zobrazuje rezervace přehlednějším způsobem. Každá událost v kalendáři je nadepsána názvem vybavení a jménem žáka. Události jsou barevně odlišeny podle stavu (červená – čeká na schválení, zelená – schválená, oranžová – probíhá). Stav rezervace určuje, která cesta se má přiřadit pro odkázání na manipulační stránku.

Další část tvoří výpisy jednotlivých rezervací. Ty jsou realizovány prostřednictvím tabulek a jsou si všechny velmi podobné (liší se pouze v možných akcích pro každou položku a informacemi ve sloupcích). Pro dosažení jejich responzivity obsah tabulek na menších obrazovkách „přetéká“ mimo obrazovku a je možné si ho zobrazit tažením prstem nebo myší daným směrem. Každý tento výpis je stránkovaný a umožňuje vyhledávání např. pomocí názvu vybavení či jména žáka. Rezervace s nastalými problémy se označují červenou barvou.

| Admin   |                         |                      |                         |            |      |
|---|-------------------------|----------------------|-------------------------|------------|------|
| <div>Dashboard</div> <div>Kalendář</div> <div>Rezervace</div> <div>Neschválené 2</div> <div>Schválené 2</div> <div>Probíhající 2</div> <div>Archiv 2</div> <div>Ostatní</div> <div>Vybavení</div> <div>Kategorie</div> <div>Přehled akcí</div> <div>Manuál</div> <div>Teacher One</div> |                         |                      |                         |            |      |
| Neschválené žádosti   |                         |                      |                         |            |      |
| Vyhledat  |                         |                      |                         |            |      |
| #   | VYBAVENÍ                | JMÉNO ŽÁKA           | ZAČÁTEK                 | KONEC      | AKCE |
| 1   | 📷 JORY GorillaPod 1K... | František Kasík, 4.H | 14.03.2025 (před 3 dny) | 21.03.2025 | 🗑️   |
| 2   | 📷 Canon EF 75-300m...   | František Kasík, 4.H | 19.03.2025 (za 2 dny)   | 21.03.2025 | 🗑️   |

Obrázek 13: výpis neschválených žádostí

Jak bylo již zmíněno, každá rezervace má rozdílné akce podle jejího druhu. Zobrazují se v posledním sloupci „AKCE“. Dle druhu rezervace je správce přesměrován na specifickou stránku stejně jako tomu je u kalendáře. Například specifickou akcí pro neschválené rezervace je schválení či odmítnutí. Při tomto procesu může správce vzít v potaz různé faktory včetně předešlých rezervací žáka, které se dělí podle toho, zda se vážou na vybavení právě přihlášeného správce nebo ne.

|  |  |
|--|--|
| <div>Admin</div> <div>Dashboard</div> <div>Kalendář</div> <div>Rezervace</div> <div>Neschválené 2</div> <div>Schválené 2</div> <div>Probíhající 2</div> <div>Archiv 2</div> <div>Ostatní</div> <div>Vybavení</div> <div>Kategorie</div> <div>Přehled akcí</div> <div>Manuál</div> <div>Teacher One</div> | <div>Neschválené žádosti &gt; Žádost #15</div> <div>Žádost #15</div> <div>  <div> <b>Canon EF 75-300mm f/4.0 - 5.6 II...</b> <div> Kategorie: objektiv • Canon<br/> Přidáno: 16.03.2025<br/> Množství: 1 ks </div> </div> </div> <div> <div> <b>Trvání</b> <div> Začátek 19.03.2025 (za 2 dny)<br/> Konec 21.03.2025<br/> Délka 2 dny </div> </div> <div> <b>Žák</b> <div> Jméno František Kasík, 4.H<br/> Email student11@spseplzen.cz </div> </div> </div> <div> <b>Zpráva od žáka</b> <div>„Natačení dokumentu o škole“</div> </div> <div> <b>Historie žáka (2)</b> <div>REZERVACE NA VAŠE VYBAVENÍ</div> <div> <div>Přidat poznámku</div> <div> <div>Odmítnout</div> <div>Přijmout</div> </div> <div> Přidat poznámku, důvod odmítnutí ... </div> </div> </div> |
|--|--|

Obrázek 14: stránka pro schválení žádosti

Stránka „Vybavení“ nabízí přehled o spravovaném vybavení. Výpis nabízí základní možnosti filtrace a řazení. Na rozdíl od výpisu rezervací je možné provádět vybrané akce jako mazání bez nutnosti přechodu na editační formulář. Při provedení takové akce dojde k okamžitému projevení změn, při kterém se zachovává pozice na stránce i přesto, že došlo k obnovení dat

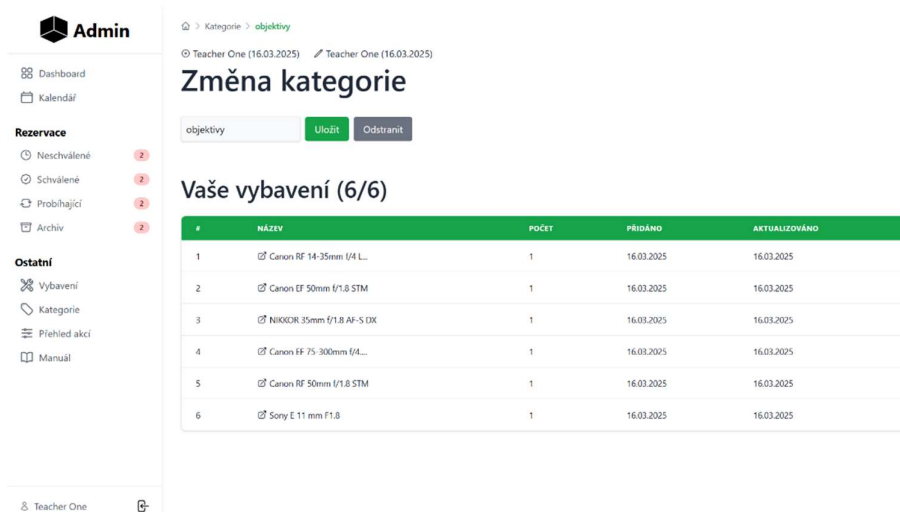


na serveru a jejich zpětnému poslání uživateli. Toto je příklad práce nástroje Inertia. Destruktivní akce typu mazání jsou opatřeny vyskakovacími okny vybízejícími k potvrzení. Tím se zabráňuje možnosti nechtěného provedení akce. Nejnáročnější funkcí na implementaci pro vybavení bylo tzv. dočasné mazání (popsáno v uživatelském manuálu), jelikož se musely ošetřit specifické krajní případy. Takovým může být archivovaná rezervace vázaná na smazané vybavení. V tomto případě je žákovi znemožněno přejít na rezervační stránku vybavení.

| # | NAZEV            | KATEGORIE | POČET | REZERVACE | PRIDÁNO    | AKTUALIZOVÁNO | AKCE |
|---|------------------|-----------|-------|-----------|------------|---------------|------|
| 1 | Canon EF 50mm... | objektiv  | 1 ks  | 2         | 16.03.2025 | 16.03.2025    |      |
| 2 | Canon EF 75mm... | objektiv  | 1 ks  | 2         | 16.03.2025 | 16.03.2025    |      |
| 3 | Canon R9 14mm... | objektiv  | 1 ks  | 1         | 16.03.2025 | 16.03.2025    |      |
| 4 | Canon RF 50mm... | objektiv  | 1 ks  | 0         | 16.03.2025 | 16.03.2025    |      |
| 5 | Eternico Mini... | stativ    | 1 ks  | 0         | 16.03.2025 | 16.03.2025    |      |
| 6 | JOBY Gorilla...  | stativ    | 1 ks  | 1         | 16.03.2025 | 16.03.2025    |      |
| 7 | Kingjoy VT-860   | stativ    | 1 ks  | 0         | 16.03.2025 | 16.03.2025    |      |
| 8 | Sony E 11 mm...  | objektiv  | 1 ks  | 0         | 16.03.2025 | 16.03.2025    |      |

Obrázek 15: stránka Vybavení

Poslední význačnou stránkou je „Kategorie“. Ta se od zbytku panelu liší tím, že kategorie jsou sdíleny mezi všemi správci, aby nedocházelo k zbytečným pokusům o vytvoření duplicitní kategorie. To však znamená, že každý správce může zasahovat do cizích kategorií. Lze však pouze mazat kategorie, které se nevyužívají. Na detailu každé kategorie může správce vidět své vlastní vybavení s danou kategorií a dále jméno správce, který kategorie vytvořil a který ji naposledy upravil.



Obrázek 16: detail kategorie

### 3.2 Nástroj Inertia.js

Inertia.js představuje jeden z nejzásadnějších nástrojů celého projektu, bez kterého bychom se neobešli. Zajišťuje chování, které je typické pro jednostránkové webové stránky (SPA), což je plynulý přechod mezi stránkami bez načtení okna prohlížeče.

V případě Inertia dojde při prvním dotazu ke klasickému načtení celé HTML stránky s veškerým JavaScriptem a CSS styly. Každý další dotaz na server však následně nezískává kompletní stránky, ale pouze jejich obsah ve formátu JSON. Součástí každé odpovědi ze serveru je název komponenty stránky a data, která se jí předají k vykreslení.

Výhodou vedle plynulosti je, že vše má na starosti v našem případě Laravel – routování, autentikace, autorizace, zabezpečení a mnoho dalšího. Ve stejný moment se kód frontendu i backendu nachází ve stejném adresáři. Vedle toho Inertia poskytuje řadu užitečných nástrojů, které dělají vývoj stránek nejen jednodušším, ale i záživnějším.

### 3.3 Adresářová struktura

Veškerý kód frontendu je uložen ve složce resources. V té se vyskytují další podsložky, kterými jsou: css (pro styly aplikace), js (komponenty, layouty a šablony stránek) a views (neměnné soubory nezbytné pro vykreslení stránek).

Vedle složky resources s frontendem souvisí konfigurační soubory nacházející se přímo v kořenovém adresáři. Těmito soubory jsou tailwind.config.js (konfigurace stylů), vite.config.js (konfigurace vývojářského serveru, sestavení, pluginů atd.), postcss.config.js (správa a konfigurace pluginů pro CSS) a .prettierrc (konfigurace formátovače kódu).

### 3.3.1 Soubor app.css

Složka css obsahuje veškeré styly aplikace. Uvnitř se nachází jediný soubor app.css. Na počátku souboru jsou importovány veškeré třídy frameworku TailWind.

Vedle toho jsou součástí souboru i vlastní třídy, které byly vytvořeny pro opakující se kombinace stylů např. pro tlačítka či formulářové prvky. Můžete si všimnout i direktiv @layer, které vytvářejí vrstvy. Ty rozdělují třídy do různých kategorií a starají se o to, aby nedocházelo k překrývání mezi různými styly. Celkem existují 3 vrstvy, do kterých lze vlastní styly napsat. Samozřejmě není nutné se vrstev držet, ale doporučuje se to.

První vrstva „base“ je určena k nastýlování základních prvků, které představují např. nadpisy, obrázky a paragrafy. Další vrstvou je „utilities“. Do té se řadí jednoúčelové třídy, které dělají pouze 1 činnost. Často slouží pro udávání velikosti písma či barvy pozadí apod. Poslední vrstva se nazývá „components“. Jak z názvu vyplývá, zde se nacházejí vlastní styly pro komponenty. Komponentou můžeme rozumět třeba tlačítko či kartu produktu atd.

### 3.3.2 Kořenový soubor

Kořenovým souborem, jak tomu u React aplikací bývá, je soubor app.jsx, který se nachází ve složce „js“. V souboru jsou importovány již zmíněné styly aplikace, soubory nezbytné pro knihovny a layouty pro správce a žáka.

Zde však standardní struktura souboru „app.jsx“ pro běžné React aplikace končí. Následuje struktura specifická pro nástroj Inertia, o kterém se můžete dozvědět přesnější informace z oficiální dokumentace. Ve zkratce je ve zbylé části souboru app.jsx určováno, která stránka se má načíst, jaký má být titulek záložky v prohlížeči a který layout se má aplikovat. Pokud má stránka definovaný layout, který má používat, tak je použit ten. V opačném případě je layout vybrán na základě role přihlášeného uživatele.

To, že je soubor „app.jsx“ adresářovým souborem, není však v tomto případě zcela pravdivé. Opravdovým kořenovým souborem je „app.blade.php“ ve složce „views“. Jeho struktura je určena nástrojem Inertia a nedoporučuje se ji měnit.

Vedle souboru „app.jsx“ se ve stejné složce nachází soubor „ziggy.js“. Ten se opět nedoporučuje měnit, ale je dobré vědět, že je nezbytný pro správné routování na straně frontendu a je generován knihovnou Ziggy. Díky této knihovně se na cesty v routeru nemusíme na frontendu odkazovat přes URL adresu, ale pomocí jména cesty. To značně napomáhá udržitelnosti aplikace a bude později vysvětleno.

### 3.3.3 Komponenty

Pro každý prvek, který se v aplikaci opakuje, nebo by se opakovat mohl, byla vytvořena vlastní komponenta. Všechny komponenty jsou umístěny ve složce „Components“. Tato složka se dále větví na další podsložky, které byly vytvořeny pouze v případě, kdy existovalo více komponent sdílejících podobnou funkci. Příkladem je složka „Navigation“ obsahující komponenty sloužící pro pohyb po stránce pomocí odkazů.

Při importování komponenty do stránky není nutné uvádět celou relativní adresu. Nástroj Vite umožňuje vytvořit tzv. alias, což je ve zkratce zástupný znak. V tomto případě je `@` zástupným znakem pro adresu `/resources/js` a je definován v souboru `vite.config.js`. V praxi to znamená pohodlnější import komponent a jiných závislostí.

Ve specifických případech, kdy se nějaká komponenta opakuje pouze v rámci 1 stránky, není pro ni vytvořen vlastní soubor ve složce „Components“ a nachází se ve stejném souboru jako stránka.

### 3.3.4 Layouty

Všechny používané layouty se nachází ve složce „Layouts“. Těmi nejčastějšími jsou „AdminLayout.jsx“ a „UserLayout.jsx“, jelikož jsou automaticky aplikovány, pokud stránka přímo nedefinuje, který layout má používat.

### 3.3.5 Stránky

Stránky představují šablony, do kterých se dynamicky doplňuje obsah. Nachází se ve složce „Pages“. Ta je v základu rozdělena na podsložky podle rolí uživatelů a to „Admin“ a „Student“. Na stejné úrovni se nachází složka „Auth“, jelikož se jedná o společnou část jak pro správce, tak i pro žáky. Tato složka obsahuje stránku pro přihlašovací formulář.

Stránky jsou dále děleny do adresářů podle dat, se kterými se na nich pracuje, nebo podle logického uspořádání. Například pro stránku katalogu byla vytvořena samostatná složka, jelikož existuje stránka pro výpis všeho dostupného vybavení a dále stránka pro zobrazení detailu vybavení. Tyto dvě stránky spolu logicky souvisí, jelikož obě zobrazují vybavení. Stránky s názvem „Index.jsx“ představují hlavní stránky, ze kterých je odkazováno na ty více specifické v rámci stejné složky.

Jednoduché stránky, které se dále nijak nevětví, jsou vloženy přímo do příslušného modulu (Admin nebo Student) bez vytvoření vlastního adresáře.

## 3.4 Layouty

Jak samotné umístění, tak i základní použití layoutů bylo již zmíněno. Ted' si řekneme více o tom, jak obecně pracují a k čemu je který určený.

Layouty obsahují sdílenou funkcionalitu mezi souvisejícími stránkami a tvoří kostru stránky. Obvykle se odlišují převážně vzhledem a umístěním navigace.

### 3.4.1 UserLayout a AdminLayout

Tyto 2 layouty jsou si svou funkcionalitou podobné. Oba pracují s tzv. sdílenými daty, což jsou data, která jsou potřebná na každé stránce.

Běžnými daty v tomto kontextu rozumíme např. informace o všech kategoriích z databáze. Tuto informaci předáváme pouze stránce, která třeba potřebuje názvy kategorií pro filtrování mezi vybavením, avšak nejsou nutná pro stránku s manuálem.

Na rozdíl od toho sdílená data představují např. informace o uživateli – jeho jméno, příjmení a email. Tuto informaci obvykle potřebujeme na většině stránek, jelikož ji v našem případě zobrazujeme v hlavní navigaci uvnitř layoutu a museli bychom ji tedy pokaždé ručně získávat z databáze a předávat každé stránce.

Oba layouty se v základu liší pouze svým rozložením, kdy uživatel nemá přístup k příliš velkému množství odkazů, a tudíž se pohodlně vejdou vedle sebe do horní navigace. V případě správce by však nastal problém, kdy by došlo k „přetečení“ obsahu stránky mimo viditelnou oblast prohlížeče, což by působilo nevzhledně a porušovalo zásady responzivního designu. Proto jsou odkazy vypisovány do levého postranního sloupce pod sebe.

Aby bylo možné do budoucna snadno přidávat či odstraňovat odkazy z hlavní navigace, jsou odkazy v obou layoutech v poli, jehož obsah se pomocí smyčky vypisuje do navigace. Díky tomu není nutné procházet celé layoutové soubory a kód stačí měnit pouze na jediném místě.

### 3.4.2 NoLayout a CenteredLayout

Tyto 2 layouty jsou si opět určitým způsobem podobné. Oba se používají v rámci aplikace v případech, kdy uživatel není přihlášen, jelikož ani jeden z layoutů o něm tyto data nevypisuje, a tudíž je nepotřebuje. Zároveň se nejedná o defaultní layouty a pro jejich použití se musí manuálně nastavit.

```
Login.layout = (page) => <CenteredLayout>{page}</CenteredLayout>;
```

Obrázek 17: příklad nastavení jiného než defaultního layoutu

CenteredLayout zarovnává veškerý obsah stránky na střed a používá se např. pro přihlašovací formulář.

Jak je z názvu patrné, NoLayout nespecifikuje žádné rozložení stránky, a při jeho použití je nutné si celé rozložení stránky vytvořit ručně. V aplikaci se používá pouze pro errorovou stránku a představuje možnost, jak deaktivovat defaultní layouty.

### 3.5 Přesměrování

Klíčovou roli při přesměrování v rámci systému opět hraje Inertia, která je v základu routovací knihovnou. Stejně jako populární knihovna React Router poskytuje komponentu „Link“, která zabráňuje obnovení celé stránky při navštívení odkazu.

Má několik atributů, ale tím jediným povinným je „href“, což je samotný odkaz. Tento odkaz generujeme pomocí knihovny Ziggy. Ta poskytuje hook (speciální funkci) useRoute, který se napříč celou aplikací ukládá do proměnné s názvem „route“, na kterou budete často narážet.

Ziggy nám umožňuje se z frontendu odkazovat na cesty definované v Laravel routeru pomocí jejich jména. To je výhodné především pro lepší udržitelnost. Kdybychom chtěli např. změnit URL adresu z „/admin/dashboard“ na „/spravce/ovladacipanel“, tak nemusíme měnit všechny URL adresy v každém souboru, který s tímto odkazem pracuje. To je z toho důvodu, že samotný název cesty se nezměnil. Je však nutné po přidání/úpravě jakékoliv cesty v Laravel routeru vygenerovat seznam dostupných cest, který používá frontend, pomocí příkazu „php artisan ziggy:generate“.

Pro Link je dostupná celá řada nepovinných parametrů. Pro zachování pozice na stránce můžeme použít atributu „preserveScroll“. Tím nedojde např. při smazání vybavení k návratu na začátek stránky. Dalším užitečným atributem je „preserveState“, díky čemuž se zachová původní stav stránky, a dokonce i focus na formulářový prvek. Příkladem užití v aplikaci je vyhledávací pole.

Jednou z těch zajímavějších vlastností je „prefetch“. Ta nám umožňuje při najetí myši na odkaz po určité době přednačíst data další stránky pro okamžité načtení. Nedoporučujeme však tuto vlastnost používat pro každý odkaz z důvodu možného zaplnění cache paměti prohlížeče. Data o stránce se drží v cache po určitou dobu, což není ideální např. pro správce či rezervační kalendář, kde je třeba vždy mít co nejvíce aktuální informace.

## 3.6 Předávání dat

Veškerá data jsou z Laravelu předávána komponentě Inertia, která obaluje každou stránku. Pro jejich zobrazení pro jednodušší debugování doporučujeme použít prohlížečové rozšíření React Developer Tools, což je nezbytný nástroj pro vývoj stránek v Reactu.

Pro zobrazení všech dostupných dat na aktuální stránce přejděte do vývojářských nástrojů v prohlížeči a klikněte na záložku „Components“ s logem Reactu. Následně klikněte na komponentu „Inertia“ a v sekci nazvané hooks na pravé straně si rozbalte strom State -> page -> props pro zobrazení dostupných dat.

### 3.6.1 Z backendu na frontend

Stejně jako je tomu u frameworku Nette, Laravel v tomto případě předává data stránkám (šablonám). Celkem se předávaná data dají rozdělit na 2 kategorie.

První kategorií jsou sdílená data, která jsou dostupná automaticky na všech stránkách bez nutnosti je manuálně předávat každé stránce zvlášť. Která data jsou sdílena nelze určit na frontendu – vše je v režii Laravelu. Pro přístup ke sdíleným datům se používá tzv. usePage hook z knihovny @inertiajs/react. Ten nám dává přístup ke všem vlastnostem stránky včetně sdílených dat. S těmito daty se převážně setkáte na již popsaných layoutových stránkách.

Druhým typem jsou data specifická pro stránku. K těm lze taktéž přistupovat pomocí usePage, avšak existuje jednodušší způsob, který nevyžaduje import této funkce. Dělejme, že máme stránku s výpisem archivovaných rezervací. Laravel nám získá seznam rezervací z databáze a odešle v proměnné s názvem „archivedReservations“ stránce v adresáři „Pages“, kde se nachází komponenta Archived.jsx. Pro zpřístupnění těchto informací stačí v komponentě vykreslované stránky uvést v jejích parametrech do složených závorek proměnnou se stejným názvem. V případě většího počtu dat jsou parametry oddělovány čárkami.

```
const Archived = ({ archivedReservations }) => {
```

Obrázek 18: příklad zpřístupnění předávaných dat

### 3.6.2 Z frontendu na backend

Existuje několik způsobů, ale tím nejběžnějším je předávání dat pomocí knihovny Ziggy. Uvádí se jako 2. argument ve složených závorkách ve formátu klíč: hodnota. K předávání dat dochází i v případě formulářů, což je však kvůli své rozsáhlosti kapitola sama o sobě.

```

<Link
  href={route('admin.reservations.waiting.show', {
    id: reservation.id,
  })}
>
  <EyeIcon className="h-5 w-5" />
</Link>

```

Obrázek 19: příklad přesměrování s předáním hodnot

### 3.6.3 Formuláře

Jedná se o jeden z nejpoužívanějších prvků. Součástí opakovaně zmiňovaného nástroje Inertia je tzv. form helper (formulářový pomocník). Skrz něj je realizována většina formulářů, jelikož výrazně usnadňuje veškerou práci a přidává další funkce, které dělají formuláře bezpečnějšími a uživatelsky přívětivějšími.

Klíčovými funkcemi je zabezpečení proti CORS útokům, zabránění opětovného odeslání formuláře při jeho zpracování, zjednodušené nahrávání souborů s možností zobrazení stavu nahrávání a řada dalších funkcí. Vedle metod POST a GET jsou podporovány i metody PATCH (pro změnu existujících hodnot) a DELETE (odstranění záznamů).

Pro detailnější pochopení doporučujeme se odkázat na oficiální dokumentaci nástroje Inertia, kapitola Forms.

## 3.7 Výpis dat

Tato část dokumentace popisuje, jakým způsobem pracujeme s daty v Reactu, jak zajišťujeme bezpečnost proti XSS útokům, a jak implementujeme různé techniky pro výpis dat, podmíněné styly a další.

### 3.7.1 Výpis prostých hodnot

Pro vypsání jednoduchých datových typů jako jsou čísla či řetězce v Reactu napište danou proměnnou do složených závorek uvnitř HTML.

React automaticky chrání před XSS útoky tím, že escapuje všechny hodnoty, které jsou vkládány do DOMu. To znamená, že pokud předáváte data do atributů nebo textového obsahu, React je automaticky ošetří. Je možné tuto funkci vypnout pomocí specifické metody, avšak silně to nedoporučujeme obzvlášť u uživatelských vstupů.



### 3.7.2 Výpis polí

Pro iteraci a výpis přes pole prvků se využívá metoda `map`, kterou můžete znát z JavaScriptu. Je důležité zmínit, že každý vypisovaný HTML prvek musí být opatřen atributem `key`, jehož hodnota musí být unikátní. Proto je většina polí objektů posílaných z Laravelu opatřena `id`.

```
{primaryNavItems.map((item, index) => (  
  <li key={index}>  
    <NavLink href={item.href}>  
      <item.icon className="w-6 md:hidden" />  
      <span>{item.label}</span>  
    </NavLink>  
  </li>  
))}
```

Obrázek 20: příklad výpisu pole hodnot

### 3.7.3 Podmíněné stylování a výpis

V celé aplikaci na straně frontendu se jedná o nejběžněji používanou praktiku ze všech. Často se totiž setkáváme se situací, kdy vedle jiných CSS tříd potřebujeme prvek přidat či odebrat jiné na základě určité podmínky. Využíváme tzv. template literals pro výpis JavaScriptu uvnitř HTML. Uvnitř dle potřeby lze využít různé logické a relační operátory. Nejčastěji se však setkáte s ternárním operátorem.

```
<div className={` ${isMobileMenuOpen ? 'dark:bg-gray-800' : ''} z-50`}>  
  <div>...</div>  
</div>
```

Obrázek 21: příklad použití podmíněných stylů

## 3.8 Komponenty

Pro aplikaci byla vytvořena celá řada komponent. Nemá však smysl do detailu popisovat každou z nich kvůli jejich jednoduchosti. Proto si rozebereme pouze ty, které jsou zásadní a složitější oproti jiným

### 3.8.1 FlashMessages.jsx

Jedná se o komponentu, s kterou se setkáte v podstatě na každé stránce, jelikož je součástí všech layoutů. Jejím úkolem je informovat prostřednictvím krátké zprávy, která sama po uběhnutí určité doby zmizí. Uživatel je informován o času, který mu zbývá k přečtení, a má možnost zprávu ručně zavřít.

Pro správné fungování komponenty je předpokládáno, že stránce je pomocí sdílených dat posílán datový objekt „flash“, který obsahuje samotnou zprávu, typ zprávy (informační, varovná...) a časové razítko (timestamp). Časové razítko je důležité proto, aby nedocházelo při manuálním obnovení stránky k opětovnému zobrazení staré zprávy, která se již sama zavřela.

Kontrola toho, zda přišla nová zpráva a zda je aktuální, probíhá pomocí „useEffect“, což je speciální funkce zabudovaná do Reactu. Spouští blok kódu, když nastane změna v uvedených proměnných.

Komponenta je udělána tak, aby se dala snadno konfigurovat. Lze nastavit např. maximální stáří zprávy, čas k přečtení a plynulost animace časovače. Je možné přidávat i více typů zpráv přidáním náležitých tříd do souboru app.css ve formátu flash-progress-[typ], flash-[typ], flash-progress-[typ]. Zároveň se musí nově vytvořené třídy přidat na tzv. safelist v souboru tailwind.config.js. To je z toho důvodu, že třídy, které se dynamicky generují, jsou vedle těch nepoužívaných automaticky odstraňovány.

### **3.8.2 ConfirmModal.jsx**

Komponenta pro vyskakovací okno, které vybízí k potvrzení nějaké akce obvykle destruktivního charakteru. Opět se jedná o často se vyskytující prvek a zprvu nemusí být zcela jasné, jak funguje nebo jak ji použít.

Základ komponenty tvoří nevšední HTML element dialog. Ve skutečnosti se již jedná o samotné vyskakovací okno vybavené vši potřebnou funkcionalitou jako zarovnání na střed, zavření při stisku klávesy ESC nebo kliknutím mimo okno.

Parametr isOpen udává, zda je okno otevřené či zavřené (true/false). Dalším je onClose, což je funkce, která se spustí při zavírání okna jak pomocí klávesové zkratky, kliknutím mimo okno tak i při kliknutí na tlačítko „Zrušit“. Při potvrzení se naopak použije funkce uvnitř onConfirm. Zbylé parametry title a children představují obsah okna informující uživatele o potvrzované akci.

### **3.8.3 BookingCalendar.jsx**

Tato komponenta se používá pouze na detailní stránce vybavení na straně žáka a slouží pro vybrání data rezervace. Informuje uživatele o dostupnosti vybavení na určité dny a uvádí řadu dalších pravidel, která jsou ověřována za správce. Jedná se o implementaci knihovny React DatePicker.

Komponenta není zcela univerzální a ve skutečnosti by se o komponentu ani jednat nemuselo, avšak kód je díky tomu přehlednější. Pro stylování se používají specifické třídy pocházející

přímo z knihovny kalendáře. Pro lepší pochopení doporučujeme si pročíst oficiální dokumentaci.

Jedním z parametrů je `bookedRanges`. Je vyžadováno, aby data byla ve formátu pole objektů, kde každý objekt má začátek rezervace (`start_date`) a konec rezervace (`end_date`). Dalším parametrem je `equipment`, který se používá pouze pro získání počtu vybavení. Posledním parametrem je `daysDisabled`, což je pole čísel, kterými se deaktivují dny v každém týdnu.

Prvním významným logickým prvkem je pole `disabledDays`, které definuje nerezervovatelné (zašedlé) dny. Toto pole se skládá z `daysDisabled`, víkendů a všech dnů, které předcházejí aktuálnímu dni + 2 dny (znemožnění rezervace na stávající den a ten následující).

Tou nejdůležitější funkcí je `findBookedDates`. Ta bere pole rezervací a přemění ho podle použitého callbacku na pole dnů daného typu (plně obsazené, částečně obsazené a téměř obsazené). Celkem jsou k dispozici 3 callbacky určené k použití s touto funkcí: `isFullyBooked` (vrací pouze obsazené dny), `isPartiallyBooked` (dny s existujícími rezervacemi) a `isAlmostBooked` (dny, kde zbývá poslední kus).

### 3.8.4 Pagination.jsx

Jak název napovídá, tato komponenta slouží pro vykreslení stránkovače na stránkách, které tuto funkcionalitu podporují. Je používána tedy tam, kde se očekává výstup většího množství dat a kde je požadováno udržení vyššího výkonu. Zároveň napomáhá s orientací uživatele.

Přijímá jediný parametr, kterým jsou odkazy na stránky výstupu. O generování těchto odkazů se stará Laravel. Pokud pole odkazů odkazuje pouze 3 prvky, tak to ve skutečnosti znamená, že se zde nachází pouze 1 stránka a není nutné vykreslovat stránkování. To je z toho důvodu, že odkazy na předchozí a další stránku se do tohoto pole taktéž počítají.

Součástí je dílčí komponenta `PaginationLink` představující odkaz na specifickou stránku. Stránkování nepočítá s velkým množstvím stránek, při kterém může dojít k méně vzhlednému výpisu na menších obrazovkách

## 3.9 Možná vylepšení

Přestože jsme s finální podobou rezervačního systému spokojeni, existuje celá řada možných funkcí, kterou jsme nestihli implementovat.

### **3.9.1 Recenze**

Bylo by na místě pod každým vybavením vypsát i hodnocení všech žáků. Hodnotit by mohli pouze ti, kteří mají v historii rezervaci právě pro toto vybavení. Celkovou spokojenost uživatelů by bylo dále vhodné zobrazovat i v katalogu a přizpůsobit podle ní výpis novým filtrem.

### **3.9.2 Správa uživatelů**

Na straně administrátora chybí správa uživatelů, která by mu umožnila si zobrazit profily a případně zakázat uživateli cokoliv rezervovat po určitou dobu. To by šlo taktéž zautomatizovat tak, že po určitém počtu problémových rezervací by došlo k automatickému odepření rezervací.

### **3.9.3 Rezervace učitelů**

Současně systém neumožňuje učitelům si vytvořit rezervaci na žádné vybavení. Bylo by na místě vytvořit zvláštní sekci vybavení, která by byla přístupná pouze učitelům. Takovou rezervaci by mělo být možné vytvořit nejen na specifický den, ale i na specifickou hodinu pro využití při výuce.

### **3.9.4 Lepší časová omezení**

Při vytváření rezervace žák není informován o přesném času, do kolika hodin musí vybavení v určitý den vrátit. Pak se může stát, že přestože žák chtěl vrátit vybavení, tak nemohl, jelikož správce v ten den končí ve škole dříve než obvykle. Proto by bylo dobré vytvořit v sekci nastavení správce nějaký kalendář, ve kterém by si pro jednotlivé dny nastavil svou dostupnost. Podle toho by byla upravena rezervační pravidla v rezervačním kalendáři, kde by zároveň docházelo k prezentování detailnějších informací.

### **3.9.5 Neuvalovat postih**

Na straně administrátora chybí možnost neuvalovat žákovi postih za různé prohřešky. Vždy se může stát, že žák onemocní nebo cizí vinou dojde k rozbití vybavení. Proto by bylo vhodné z dobrých důvodů mít možnost postih neuvalit.

## 4 Backend

### 4.1 Autentizace

V této kapitole je popsán autentizační systém aplikace rezervačního systému, který je postaven na integraci se školním LDAP serverem. Implementace autentizace je klíčovou součástí celé aplikace, neboť zajišťuje bezpečný přístup uživatelů a zároveň rozlišuje mezi studenty a učiteli.

#### 4.1.1 LDAP Integrace

Pro autentizaci uživatelů byl využit školní LDAP server, což umožňuje studentům a učitelům využívat stejné přihlašovací údaje jako pro ostatní školní služby.

Ačkoliv autentizace probíhá proti LDAP serveru, potřebné údaje o uživateli jsou ukládány do tabulky v databázi. Toto řešení významně usnadňuje práci s uživatelskými daty, umožňuje rychlejší přístup k informacím a zjednodušuje implementaci vztahů mezi uživateli a dalšími entitami systému (vybavení, rezervace). Při prvním přihlášení uživatele jsou základní údaje automaticky synchronizovány z LDAP do lokální databáze.

Integrace LDAP je implementována prostřednictvím knihovny LdapRecord pro Laravel, která poskytuje způsob komunikace s LDAP serverem v rámci Laravel aplikace. Pro účely autentizace byla vytvořena třída App\Ldap\UserLocal, která rozšiřuje základní třídu uživatele z knihovny LdapRecord. Tato třída obsahuje metody pro ověření rolí uživatele na základě členství v LDAP skupinách. Metoda hasRole ověřuje, zda uživatel patří do specifické LDAP skupiny (např. 'teacher'), zatímco metoda isTeacher je specifickou implementací pro kontrolu role učitele.

Pro mapování LDAP atributů na atributy uživatelů v databázi byla vytvořena třída AttributeHandler, která zajišťuje, že je uživateli přiřazena správná role v systému na základě jeho členství v LDAP skupinách.

#### 4.1.2 Rate Limiting a ochrana přihlášení

Pro ochranu proti útokům typu brute force byl implementován rate limiting mechanismus, který omezuje počet pokusů o přihlášení. Implementace využívá vestavěnou Laravel funkcionalitu RateLimiter, která je aplikována v přihlašovací metodě LoginController.

Princip fungování rate limitingu je následující:

- Pro každou kombinaci uživatelského jména a IP adresy je vytvořen unikátní klíč
- Systém omezuje počet neúspěšných pokusů o přihlášení na 5 za minutu
- Po překročení limitu je uživateli zobrazen chybový flash message

- Po úspěšném přihlášení je limit resetován

Pro zvýšení bezpečnosti je taktéž implementována regenerace session při úspěšném přihlášení, což zabraňuje útokům typu session fixation. Po úspěšném přihlášení je uživatel přesměrován na příslušnou domovskou stránku v závislosti na jeho roli v systému – administrátoři jsou přesměrováni na dashboard, zatímco běžní uživatelé (studenti) na přehled svých aktivních rezervací.

Autentizační systém je dále podporován middlewares, které zajišťují, že uživatelé mají přístup pouze k těm částem aplikace, ke kterým mají oprávnění. To je implementováno pomocí několika middleware tříd, jako jsou Authenticated, Student a Teacher, které kontrolují stav přihlášení a roli uživatele před umožněním přístupu k jednotlivým cestám aplikace.

## 4.2 Autorizace

Po úspěšném přihlášení uživatele do systému přichází na řadu autorizace, která určuje, k jakým funkcím a datům má uživatel přístup. Implementace autorizace v rezervačním systému je postavena na rolích uživatelů, Laravel Policies a middleware mechanismech, které společně zajišťují, že každý uživatel má přístup pouze k těm funkcím a datům, ke kterým je oprávněn.

### 4.2.1 Role uživatelů (student/učitel)

Systém pracuje se dvěma základními rolemi uživatelů:

- **Student** – standardní uživatel, který má možnost prohlížet dostupné vybavení, vytvářet rezervace a spravovat své vlastní rezervace
- **Učitel** (administrátor) – uživatel s rozšířenými právy, který spravuje vybavení, schvaluje či zamítá rezervace a má přístup k administrátorskému rozhraní

Role jsou určeny na základě členství uživatele v příslušných LDAP skupinách, jak bylo popsáno v kapitole 4.1.1. V databázi je role reprezentována booleovským atributem `is_admin` v tabulce uživatelů.

Model uživatele definuje různé vztahy podle rolí. Učitel má přístup nejen ke svým rezervacím, ale také ke všem rezervacím vybavení, které spravuje. Tuto funkcionalitu zajišťuje vztah `reservationsToManage()`, který propojuje rezervace přes vybavení, které učitel vlastní.

### 4.2.2 Policies a Gates

Aplikace využívá systém Laravel Policies pro kontrolu oprávnění k jednotlivým operacím nad modely. Pro každý klíčový model (Equipment, Reservation) je vytvořena odpovídající Policy třída, která definuje pravidla přístupu.

```
public function update(User $user, Equipment $equipment)
{
    return $user->id === $equipment->user_id ? Response::allow() : Response::deny('Nemáte práva ke změně tohoto vybavení.');
```

Obrázek 22: příklad policy zabráňující změně vybavení cizích správců

Podobně funguje i `ReservationPolicy`, kde jsou definována pravidla pro operace nad rezervacemi. Základním principem je kontrola, zda uživatel:

- Je vlastníkem rezervace (v případě studenta)
- Je vlastníkem vybavení (v případě učitele)

Tyto policy třídy jsou použity v contollerech prostřednictvím `Laravel Gates`, což zajišťuje, že uživatelé mohou manipulovat pouze s těmi objekty, ke kterým mají oprávnění.

### 4.2.3 Middleware

Pro zajištění kontroly přístupu k celým sekcím aplikace jsou využity vlastní middleware třídy:

- **Authenticated** – kontroluje, zda je uživatel přihlášen
- **Student** – kontroluje, zda má přihlášený uživatel roli studenta
- **Teacher** – kontroluje, zda má přihlášený uživatel roli učitele (administrátora)
- **Guest** – kontroluje, zda uživatel není přihlášen (pro přístup k přihlašovací stránce)

Tyto middleware třídy jsou využity v definici route skupin v souboru `web.php`, kde jsou jednotlivé sekce aplikace přiřazeny příslušným middleware:

Tato struktura zajišťuje, že nepřihlášení uživatelé mají přístup pouze k přihlašovací stránce, studenti vidí pouze uživatelskou část systému a nemohou přistupovat k administraci a učitelé mají přístup ke kompletní administraci systému.

## 4.3 Zabezpečení

Zabezpečení aplikace implementuje několik vrstev ochrany zajišťujících bezpečnost uživatelských vstupů, ochranu proti běžným webovým útokům a konzistenci dat.

### 4.3.1 Validace vstupů

Validace všech uživatelských vstupů je implementována na dvou úrovních – na straně klienta pomocí `JavaScriptu` a na straně serveru pomocí `Laravel validátorů`.

Serverová validace využívá vestavěný `Laravel validační mechanismus`, který je aplikován v contollerech před zpracováním dat. Pro každý formulář v aplikaci jsou definována validační pravidla včetně chybových hlášek v českém jazyce.

Validace zajišťuje:

- Kontrolu datových typů (string, integer)
- Ověření formátu souborů a jejich velikosti
- Kontrolu rozsahu hodnot a jejich povinnosti
- Uživatelsky přívětivé chybové hlášky

Díky dvojímu ověření (klient + server) je minimalizováno riziko přijetí neplatných dat i v případě obejití klientské validace.

### 4.3.2 CSRF ochrana

Aplikace implementuje ochranu proti Cross-Site Request Forgery (CSRF) útokům pomocí standardního Laravel mechanismu. Každý formulář v aplikaci obsahuje skrytý CSRF token, který je automaticky přidáván do hlavičky HTTP požadavků odesílaných z frontend aplikace.

Laravel na straně serveru automaticky kontroluje přítomnost a platnost tohoto tokenu pro všechny POST, PUT, PATCH a DELETE požadavky, čímž brání útokům typu CSRF. Tato ochrana je implementována jako middleware, který ověřuje, že všechny příchozí neGET požadavky obsahují platný token.

Inertia.js, který je využíván pro propojení frontendu a backendu, zajišťuje automatickou správu a předávání CSRF tokenů mezi React komponentami a serverem, což zjednodušuje implementaci této ochrany v celé aplikaci.

### 4.3.3 Transakce a konzistence dat

Pro zajištění konzistence dat při složitějších operacích využívá aplikace databázové transakce. Typickým příkladem je vytváření rezervací, kde je třeba zajistit, že jedno vybavení nebude ve stejný čas rezervováno dvěma uživateli.

Tento proces zahrnuje několik kroků:

- Zahájení transakce pomocí `DB::beginTransaction()`
- Kontrola dostupnosti vybavení v požadovaném časovém období
- Vytvoření rezervace v databázi
- Odeslání notifikace uživateli
- Potvrzení transakce pomocí `DB::commit()`

V případě jakéhokoliv selhání během těchto kroků je celá transakce vrácena zpět pomocí `DB::rollBack()`, což zajišťuje, že databáze zůstane v konzistentním stavu.



Aplikace také implementuje logiku pro kontrolu překrývajících se rezervací, aby bylo zajištěno, že počet současných rezervací pro dané vybavení nepřekročí dostupné množství. Tato kontrola probíhá jak při vytváření nových rezervací, tak při změně množství kusů existujícího vybavení.

Během kontroly aplikace analyzuje všechny existující rezervace a porovnává je s požadovaným termínem. Pokud by nová rezervace nebo změna množství vybavení vedla k situaci, kdy by počet rezervací překročil dostupné množství, je operace zamítnuta a uživatel je informován o nemožnosti provedení požadované akce.

## **4.4 Správa dat**

Aplikace uchovává a zpracovává data o vybavení, uživateli, kategoriích a rezervacích v relační databázi MySQL. V této kapitole je popsána struktura dat a způsoby, jakými s nimi aplikace pracuje.

### **4.4.1 Struktura modelů a vztahů**

Aplikace využívá objektově-relační mapování prostřednictvím Laravel Eloquent, což umožňuje práci s databázovými záznamy jako s objekty. Databázový model je navržen tak, aby optimálně reprezentoval doménové entity systému a vztahy mezi nimi.

Základní entity systému a jejich vztahy:

- User – reprezentuje uživatele systému (student nebo učitel)
- Equipment – představuje jednotlivé vybavení dostupné k zapůjčení
- Category – kategorizuje vybavení do logických skupin
- Reservation – uchovává informace o rezervacích vybavení

Tyto entity jsou propojeny následujícími vztahy:

- User může vlastnit více kusů Equipment (vztah 1:N)
- User může vytvořit více Reservation (vztah 1:N)
- Equipment patří do jedné Category (vztah N:1)
- Equipment může mít více Reservation (vztah 1:N)

### **4.4.2 Soft Delete a archivace dat**

Systém implementuje mechanismus tzv. "soft delete" pro záznamy vybavení. To znamená, že při smazání vybavení nedochází k jeho fyzickému odstranění z databáze, ale pouze k označení záznamu jako smazaný (pomocí vyplnění sloupce `deleted_at`).

Tento přístup přináší několik výhod:

- Zachování historických dat a možnost obnovení smazaného vybavení
- Zachování integrity dat, u již existujících rezervací
- Možnost zobrazit smazané vybavení v administraci s možností jeho obnovení

V controllerech je pak možné pracovat jak s aktivními záznamy, tak i se smazanými.

Pro rezervace je implementován vlastní systém archivace. Ukončené rezervace nejsou smazány, ale přesunuty do stavu "archivováno", kde jsou uchovávány pro pozdější reference a statistické účely. Archivované rezervace jsou automaticky fyzicky odstraněny z databáze po uplynutí 4 let, což zajišťuje cron úloha nastavená v souboru Kernel.php.

Tento přístup umožňuje udržet databázi v rozumné velikosti a zároveň uchovávat relevantní historická data.

```
$schedule->command('reservations:delete-old-archived')
->dailyAt('00:01');
```

*Obrázek 23: cron úloha pro mazání archivovaných rezervací*

#### 4.4.3 Filtrace a vyhledávání

Uživatelé mohou filtrovat vybavení podle různých kritérií, jako jsou kategorie, výrobce, místnost nebo dostupnost. Implementace filtrace v controllerech využívá podmíněné sestavování dotazů pomocí Eloquent query builderu. Na základě poskytnutých filtrů (např. kategorie nebo vyhledávání) jsou k základnímu dotazu dynamicky přidávány odpovídající whereHas nebo where podmínky, které omezují výsledky podle zvolené kategorie nebo hledaného textového řetězce v názvu či popisu vybavení.

Tento přístup umožňuje dynamické sestavování komplexních SQL dotazů na základě uživatelských vstupů, což výrazně zlepšuje uživatelský zážitek a usnadňuje práci s velkým množstvím dat.

Pro efektivní prohledávání textových polí (název, popis) je implementováno vyhledávání využitím operátoru LIKE, který umožňuje hledat částečné shody. Vyhledávací dotazy jsou optimalizovány pomocí správného indexování databázových tabulek, což zajišťuje rychlou odezvu i při větším objemu dat.

Kromě filtrace na straně serveru je implementována i možnost řazení výsledků podle různých kritérií, jako je název, datum vytvoření nebo oblíbenost (počet rezervací). Toto řazení je implementováno pomocí metody orderBy.

## 4.5 Rezervační logika

Rezervační logika je jádrem celého systému a zajišťuje správu životního cyklu rezervací od jejich vytvoření až po archivaci. Implementace zohledňuje specifické požadavky školy na proces vypůjčování vybavení.

### 4.5.1 Životní cyklus rezervace

Rezervace prochází několika stavy, které jsou reprezentovány hodnotou atributu status v databázi:

- neschváleno – počáteční stav po vytvoření žádosti o rezervaci
- schváleno – stav po schválení rezervace správcem vybavení
- probíhá – aktivní stav po vyzvednutí vybavení
- archivováno – konečný stav po vrácení vybavení

V kontroleru `ReservationsController` je implementována logika pro přechody mezi jednotlivými stavy. Při změně stavu systém provádí validace a kontroly k zajištění integrity dat.

### 4.5.2 Kontrola dostupnosti vybavení

Klíčovou součástí rezervačního systému je kontrola dostupnosti vybavení v požadovaném časovém období. Pro každou novou rezervaci je potřeba ověřit, zda je požadované vybavení v daném termínu k dispozici.

Metoda `checkAvailability` v třídě `EquipmentController` analyzuje existující rezervace pro dané vybavení a vrací jeden ze tří stavů dostupnosti: plně dostupné (hodnota 1), částečně rezervované (hodnota 2), plně nedostupné (hodnota 3).

### 4.5.3 Zpracování kolizí rezervací

Systém musí zabránit situacím, kdy by více uživatelů rezervovalo stejné vybavení ve stejném čase nad rámec dostupného množství. Pro zajištění konzistence dat při souběžných rezervacích je implementována databázová transakce.

Při vytváření rezervace je nejprve zahájena transakce a zkontrolována dostupnost vybavení. Pokud není dostupné, transakce je okamžitě vrácena zpět (rollback). V opačném případě je vytvořen záznam rezervace a transakce je potvrzena (commit). V případě jakékoliv chyby během procesu je transakce automaticky vrácena zpět, čímž se zajistí integrita dat.

#### 4.5.4 Automatizované akce

Rezervační systém obsahuje několik automatizovaných akcí, které se pravidelně provádějí a zajišťují efektivní správu rezervací:

- Neschválené rezervace – Rezervace, které nebudou včas schváleny, budou automaticky odstraněny. Tato kontrola probíhá každý den.
- Nevyzvednuté rezervace – Vybavení, které si žák včas nevyzvedne, bude odstraněno, aby si ho mohli rezervovat ostatní. Systém tuto kontrolu provádí každý den.
- Historie rezervací – Historie rezervací bude promazána každé 4 roky, aby nepřekážely v databázi. Tím je zajištěno, že databáze nebude zbytečně narůstat a zůstane výkonná.

Tyto akce jsou implementovány pomocí Laravel Command tříd a spouštěny pomocí Laravel Task Scheduleru, který je konfigurován v souboru `app/Console/Kernel.php`.

Důležitou součástí systému je také validace dat rezervace. Systém ověřuje, že rezervace nezačíná ani nekončí o víkendu, a že začátek rezervace je alespoň jeden den po jejím vytvoření, aby měl správce dostatek času na schválení.

### 4.6 Notifikační systém

Rezervační systém implementuje notifikační mechanismy pro informování uživatelů o změnách stavu rezervací a důležitých událostech.

#### 4.6.1 Emailové notifikace

Aplikace využívá Laravel Notification systém pro zasílání emailových notifikací uživatelům. Notifikace jsou implementovány jako samostatné třídy v adresáři `app/Notifications` a implementují rozhraní `ShouldQueue` pro asynchronní zpracování.

Systém zasílá následující typy emailových notifikací:

- `ReservationReceived` – informuje uživatele o přijetí nové žádosti o rezervaci
- `ReservationApproved` – informuje uživatele o schválení rezervace
- `ReservationDisapproved` – informuje uživatele o zamítnutí rezervace
- `ReservationCancelledAlert` – informuje správce o zrušení rezervace uživatelem
- `ReservationReminder` – připomíná uživateli blížící se konec rezervace

Odeslání notifikace je implementováno pomocí fasády `Notification`.

## 4.6.2 Flash messages

Pro informování uživatelů o výsledku akcí prováděných v systému jsou implementovány Flash messages. Tyto zprávy jsou dočasně uloženy v session a zobrazeny uživateli po přesměrování. Systém implementuje vlastní helper funkci `flash()`, která vytváří strukturovaná data pro flash zprávy.

Flash zprávy jsou použity po celém systému pro informování o výsledcích akcí a jsou automaticky sdíleny s frontend částí aplikace prostřednictvím middleware `HandleInertiaRequests`, který je zaregistrován v souboru `app/Http/Kernel.php`.

## 4.7 Optimalizace výkonu

Backend aplikace obsahuje několik optimalizačních technik, které zajišťují rychlou odezvu i při větším zatížení systému.

### 4.7.1 Zpracování obrázků

Pro efektivní práci s obrázky vybavení je implementována asynchronní úprava a optimalizace pomocí třídy `ProcessEquipmentImage`, která běží jako queue job. Metoda `handle` této třídy načte nahraný obrázek, optimalizuje jej (změna velikosti, převod do WebP), uloží do veřejného úložiště a aktualizuje cestu v databázovém záznamu daného vybavení. Celý proces zahrnuje logování případných chyb a následné smazání dočasného souboru.

Implementované optimalizace obrázků:

- Převod do formátu WebP pro minimalizaci velikosti souborů
- Automatické změnění velikosti obrázků na šířku 800px
- Uložení do organizované adresářové struktury
- Čištění dočasných souborů po zpracování

### 4.7.2 Queue Jobs

Aplikace využívá Laravel Queue systém pro asynchronní zpracování časově náročných operací. Časově náročné úlohy, které by zpomalovali běh aplikace, se přesouvají do fronty a jejich zpracování čeká na okamžik, kdy je aplikace méně vytížená.

Výhody použití front úloh:

- Zpracování obrázků neblokuje hlavní aplikační vlákno
- Emailové notifikace jsou odesílány asynchronně
- Lepší výkon při špičkách zatížení systému

- Možnost opakování u selhaných úloh

Všechny notifikační třídy implementují rozhraní `ShouldQueue`, což zaručuje jejich asynchronní zpracování.

```
ProcessEquipmentImage::dispatch($equipment->id, $fullTempPath, $filename, $tempPath);
```

*Obrázek 24: přidání obrázku do fronty*

### 4.7.3 Cachování a lazy loading

Systém implementuje několik technik pro minimalizaci počtu a složitosti databázových dotazů:

- Eager loading – Při načítání dat z databáze jsou vztahy načítány pomocí metod `with()`, což eliminuje N+1 query problém.
- Lazy loading kolekcí – Metoda `through()` v kombinaci s `defer()` odkládá zpracování kolekcí na straně `Inertia.js`.
- Paginace – Pro omezení množství dat přenášovaných na klienta je implementována paginace výsledků.
- Indexování databáze – Databázové tabulky obsahují indexy na často používaných vyhledávacích sloupcích a cizích klíčích, což zrychluje vyhledávací a spojovací operace.

## 4.8 Inertia.js integrace

Rezervační systém využívá `Inertia.js` jako spojovací vrstvu mezi `Laravel` backendem a `React` frontendem. Tato architektura kombinuje výhody tradiční serverové aplikace s moderním reaktivním UI.

### 4.8.1 Komunikace s frontendem

`Inertia.js` umožňuje implementovat SPA (Single Page Application) bez nutnosti vytvářet samostatné REST API. Komunikace probíhá pomocí standardních HTTP požadavků, které vracejí JSON odpovědi namísto HTML.

Backend (`Laravel` controller) využívá metodu `Inertia::render()` pro určení, která frontendová komponenta (v tomto případě `Admin/Equipment/Index`) se má zobrazit. Spolu s názvem komponenty se předávají potřebná data jako props – například seznam vybavení (`equipment`), počet smazaných položek (`trashedCount`), seznam kategorií (`categories`) a aktuální filtry (`filters`). Frontendová komponenta pak tato data přijme a vykreslí uživatelské rozhraní.

Pro routování je využíván Ziggy, který umožňuje použít Laravel routovací systém přímo v JavaScriptu. Potřebná data pro Ziggy (aktuální URL, port a seznam pojmenovaných rout) jsou sdílena s frontendem prostřednictvím metody share v HandleInertiaRequests middleware, kde jsou vložena do globálně dostupného objektu ziggy.

#### 4.8.2 Předávání dat

Inertia.js zajišťuje efektivní předávání dat mezi backendem a frontendem. Data jsou strukturována tak, aby minimalizovala množství přenášených informací. Pro zpracování velkých datových sad je implementována optimalizace pomocí metody defer(). Sdílení globálních dat pro všechny stránky je implementováno v middleware HandleInertiaRequests.

Propojení formulářů s backendem zajišťuje Inertia.js Form Helper, který automaticky řeší odesílání validačních chyb a CSRF ochranu.

### 4.9 Testování

V této části je popsána implementace testovacích mechanismů, které pomáhají odhalit případné problémy a zajistit stabilitu systému.

#### 4.9.1 Unit testy

Unit testy se zaměřují na testování jednotlivých komponent a metod v izolaci. Aplikace obsahuje několik typů unit testů:

- Testy modelů: CategoryTest, EquipmentTest, ReservationTest a UserTest ověřují správnou funkčnost datových modelů, jejich vztahů a business logiky.
- Testy helperů: FlashHelperTest a ReservationHelperTest kontrolují pomocné funkce aplikace, jako je vytváření flash zpráv a manipulace s rezervačními daty.

Unit testy využívají vestavěné testovací nástroje frameworku Laravel a jsou navrženy tak, aby běžely nezávisle na sobě a neovlivňovaly produkční data.

#### 4.9.2 Feature testy

Feature testy ověřují funkčnost celých komponent systému a jejich vzájemnou integraci. Tyto testy jsou rozděleny do několika kategorií:

- Testy administrátorských controllerů: CategoriesControllerTest, DashboardControllerTest, EquipmentControllerTest a ReservationControllerTest zajišťují správnou funkci administrátorského rozhraní.

- Testy autentizace: LoginControllerTest ověřuje proces přihlášení, integraci s LDAP a bezpečnostní mechanismy.
- Testy uživatelských controllerů: EquipmentAvailabilityTest, EquipmentControllerTest, ReservationsAdvancedTest a ReservationsControllerTest pokrývají uživatelské funkce jako je prohlížení katalogu, vytváření rezervací a kontrola dostupnosti.

Feature testy simulují HTTP požadavky na aplikaci a kontrolují, zda je vrácena očekávaná odpověď. Toto umožňuje testovat aplikaci podobným způsobem, jakým by ji používal reálný uživatel.

Příkladem testovaných funkcí je kontrola dostupnosti vybavení v konkrétních termínech, správné zobrazení formulářů pro vytvoření rezervace, schvalovací proces a automatické akce při změně stavu rezervace.

Implementované testy pokrývají kritické části aplikace, zejména rezervační logiku, autentizaci a autorizaci, zpracování dat a správu vybavení. Tato testovací strategie pomáhá zajistit spolehlivost a stabilitu systému.



## 5 NAsazení a zabezpečení

### 5.1 Deployment na Linux server

Nasazení aplikace probíhá na Linux server pomocí následujícího postupu:

1. Klonování Git repozitáře - stažení zdrojového kódu z centrálního úložiště
2. Konfigurace prostředí - vytvoření a úprava produkčního konfiguračního souboru
3. Nastavení oprávnění - zajištění správných oprávnění pro soubory a adresáře
4. Sestavení a spuštění Docker kontejnerů - kompilace aplikace a spuštění produkčního prostředí

Tento proces je navržen tak, aby byl jednoduchý a opakovatelný, což umožňuje rychlé a spolehlivé nasazení i aktualizace aplikace.

### 5.2 Zabezpečení aplikace

V rámci vývoje a nasazení rezervačního systému byla věnována značná pozornost zabezpečení aplikace na různých úrovních. Tato část dokumentace popisuje implementovaná bezpečnostní opatření, která zajišťují ochranu dat, uživatelů a systémových prostředků.

#### 5.2.1 Zabezpečená komunikace pomocí HTTPS

Pro zajištění bezpečné komunikace mezi klienty a serverem jsme zajistili protokol HTTPS s využitím self-signed certifikátu:

- Samopodepsaný SSL certifikát vygenerovaný pomocí OpenSSL s platností 365 dní
- Konfigurace HTTPS v Nginx webovém serveru
- Vynucení HTTPS přesměrováním veškerého HTTP provozu na zabezpečenou verzi

Certifikát byl vygenerován příkazem dostupným v README.

Tato implementace zajišťuje šifrovaný přenos dat mezi klienty a serverem, což efektivně zabraňuje odposlouchávání komunikace a útokům typu man-in-the-middle. Přestože samopodepsaný certifikát může v prohlížeči generovat varování o nedůvěryhodném certifikátu, pro účely školní aplikace v interním prostředí poskytuje dostatečnou úroveň zabezpečení bez nutnosti platit za komerční certifikáty nebo implementovat Let's Encrypt.

### 5.2.2 Implementace Content Security Policy

Pro ochranu před útoky typu Cross-Site Scripting (XSS) a jinými útoky založenými na injekci obsahu byl v konfiguraci webového serveru Nginx nastaven Content Security Policy (CSP):

- Omezuje načítání skriptů pouze z důvěryhodných zdrojů
- Zabraňuje vkládání aplikace do cizích rámců (ochrana proti clickjacking)
- Definuje povolené zdroje pro styly, fonty a obrázky
- Kontroluje, odkud mohou být odesílány formuláře

Implementace CSP výrazně zvyšuje bezpečnost aplikace před různými typy útoků na straně klienta.

### 5.2.3 Firewall

Na úrovni operačního systému byl nakonfigurován firewall (UFW - Uncomplicated Firewall), který omezuje síťovou komunikaci pouze na nezbytné porty:

#### Základní služby

- SSH (port 22)
- HTTP (port 80) - pro přesměrování na HTTPS
- HTTPS (port 443)

#### Monitorovací nástroje s omezeným přístupem

- Grafana
- Prometheus
- Uptime Kuma
- Portainer

Toto nastavení minimalizuje útočnou plochu systému a zabraňuje neoprávněnému přístupu k službám, které by neměly být veřejně dostupné.

### 5.2.4 Autentizace a autorizace

Systém autentizace je integrován se školním LDAP serverem:

- Centralizovaná správa uživatelů s jednotným přihlášením
- Bezpečné ukládání a ověřování LDAP přihlašovacích údajů
- Granulární systém oprávnění pro různé role (student, učitel, administrátor)

Tato integrace zajišťuje, že přístup k funkcím aplikace je správně řízen na základě rolí uživatelů, a umožňuje jednotnou správu identit napříč školními systémy.

### 5.2.5 Bezpečnostní audit Docker kontejnerů

V rámci procesu zabezpečení byl proveden bezpečnostní audit Docker kontejnerů pomocí nástroje Docker Bench Security, který identifikoval potenciální rizika a oblasti pro zlepšení:

- Kontrola konfigurace Docker démona
- Audit nastavení kontejnerů a omezení jejich oprávnění
- Kontrola správné izolace a nastavení sítě
- Ověření bezpečnosti obrazů kontejnerů

Na základě výsledků auditu byla implementována vybraná doporučení s přihlédnutím k požadavkům na funkčnost v rámci školního prostředí.

### 5.2.6 Konfigurace produkčního prostředí

Produkční prostředí bylo konfigurováno s důrazem na bezpečnost:

- APP\_DEBUG=false pro prevenci úniku citlivých informací při chybách
- Oddělené .env soubory pro vývojové a produkční prostředí
- Minimální oprávnění pro přístup ke konfiguračním souborům

Tato opatření minimalizují riziko zneužití aplikace a zajišťují, že v produkčním prostředí nejsou dostupné ladící informace, které by mohly být využity potenciálními útočníky.

### 5.2.7 Zálohování a redundance dat

Součástí bezpečnostní strategie je také pravidelné zálohování dat:

- Automatizované zálohy každé dva týdny včetně databáze a konfigurace
- Více úrovní záloh (lokální, cloudové)
- Rotace záloh s automatickým odstraňováním starších verzí

Tento přístup zajišťuje, že i v případě selhání hardware, chyby v konfiguraci nebo jiných problémů je možné rychle obnovit funkčnost systému s minimální ztrátou dat.

### 5.2.8 Shrnutí bezpečnostních opatření

Implementovaná bezpečnostní opatření vytváří komplexní ochranný systém, který efektivně minimalizuje riziko úspěšného útoku na aplikaci a zajišťuje důvěrnost, integritu a dostupnost dat v rezervačním systému. Zároveň byla identifikována některá potenciální rizika a oblasti pro budoucí zlepšení, což ukazuje na kontinuální přístup k bezpečnosti.

Přestože v rámci školního prostředí nebyla implementována všechna možná bezpečnostní opatření, která by byla nezbytná pro vysoce citlivé produkční systémy, zvolený přístup poskytuje vyváženou kombinaci zabezpečení a použitelnosti odpovídající účelu aplikace.

## 5.3 Monitoring

Tato část dokumentace popisuje implementaci a konfiguraci monitorovacího systému pro rezervační aplikaci běžící v Docker kontejnerech. Monitorovací řešení je založeno na třech klíčových nástrojích: Prometheus pro sběr metrik, Grafana pro vizualizaci a Kuma pro monitorování dostupnosti.

Implementovaný monitorovací systém poskytuje komplexní přehled o výkonu, dostupnosti a zdraví aplikace v reálném čase, což umožňuje včasnou detekci problémů a proaktivní správu infrastruktury.

### 5.3.1 Architektura monitorovacího systému

Monitorovací systém je navržen jako několik propojených komponent, které společně vytvářejí komplexní řešení pro sledování různých aspektů aplikace a infrastruktury:

- Aplikační stack (Laravel, Nginx, MySQL) generuje metriky o svém výkonu a stavu
- Specializované exportéry sbírají tyto metriky z jednotlivých komponent
- Prometheus ukládá a zpracovává shromážděné metriky
- Grafana vizualizuje data v podobě dashboardů a grafů
- Kuma sleduje dostupnost aplikace a kritických endpointů

Tato architektura umožňuje získat ucelený pohled na výkon celého systému a zároveň detailní informace o jednotlivých komponentách.

### 5.3.2 Popis komponent

1. Prometheus:
  - Časosběrná databáze optimalizovaná pro metriky
  - Primární sběrač dat (scraper) zodpovědný za pravidelné čtení metrik
  - Poskytuje PromQL (Prometheus Query Language) pro dotazování na data
  - Ukládá historická data a poskytuje základní výstrahy

## 2. Grafana:

- Vizualizační platforma pro metriky
- Umožňuje vytváření komplexních dashboardů
- Podporuje různé zdroje dat (Prometheus, MySQL, atd.)
- Poskytuje pokročilý systém výstrah a notifikací

## 3. Kuma:

- Jednoduchý, ale efektivní nástroj pro monitorování dostupnosti (uptime)
- Monitoruje dostupnost HTTP/HTTPS endpointů
- Zaznamenává odezvy, výpadky a další statistiky
- Generuje statusovou stránku

### 5.3.3 Implementace monitorovacího řešení

Monitoring naší aplikace je implementován jako součást Docker infrastruktury pomocí samostatných kontejnerů propojených s hlavní aplikací. Tento přístup zajišťuje izolaci monitorovacích nástrojů od samotné aplikace a zároveň umožňuje jejich snadnou správu a škálování.

Monitoring využívá několik klíčových nástrojů:

- Prometheus běží jako samostatný kontejner a v pravidelných intervalech (15 sekund) sbírá metriky z různých zdrojů. Jeho konfigurace definuje, odkud a jak často se mají metriky získávat.
- Node Exporter sbírá metriky na úrovni operačního systému (vyžití CPU, paměti, disku), což nám poskytuje základní přehled o zdraví hostitelského serveru.
- cAdvisor je specializovaný na sběr metrik o Docker kontejnerech, což nám umožňuje sledovat výkon a využití zdrojů jednotlivými komponenty aplikace.
- Grafana poskytuje uživatelské rozhraní pro vizualizaci dat, které jsou uloženy v Prometheus. Díky tomu můžeme vytvářet přehledné dashboardy s grafy a tabulkami.
- Uptime Kuma doplňuje monitoring o sledování dostupnosti a odezvy klíčových endpointů aplikace.

Prometheus je nakonfigurován tak, aby sbíral metriky z několika zdrojů:

- Samotný Prometheus (self-monitoring)
- cAdvisor (metriky Docker kontejnerů)
- Node Exporter (systémové metriky)
- Nginx Exporter (metriky webového serveru)
- MySQL Exporter (metriky databáze)
- PHP-FPM Exporter (metriky PHP)

#### **5.3.4 Nastavení exportérů pro sběr metrik**

Pro komplexní monitoring aplikace používáme specializované exportéry, které zajišťují sběr detailních metrik z různých komponent systému:

1. MySQL Exporter poskytuje podrobné informace o výkonu databáze:
  - Počet a typ dotazů
  - Využití připojení
  - Velikost a využití cache
  - Latence dotazů
2. Nginx Exporter sleduje výkon webového serveru:
  - Počet požadavků
  - Doba odezvy
  - Chybové kódy a jejich četnost
  - Aktivní připojení
3. PHP-FPM Exporter monitoruje backendovou část aplikace:
  - Využití PHP procesů
  - Doba zpracování požadavků
  - Počet zpracovaných požadavků

- Velikost paměti využívané PHP procesy

Tyto exportéry jsou nastaveny tak, aby automaticky sbíraly relevantní metriky a poskytovaly je Prometheus v standardizovaném formátu.

### 5.3.5 Konfigurace Grafany

Grafana je nakonfigurována s několika specializovanými dashboardy, které poskytují přehled o různých aspektech aplikace:

1. Systémový přehled:
  - Celkové využití CPU, paměti a disku
  - Síťový provoz
  - Základní zdravotní metriky
2. Docker kontejnery:
  - Využití zdrojů jednotlivými kontejnery
  - Stav kontejnerů
  - Paměťové limity a využití
3. Aplikační výkon:
  - Počet požadavků za sekundu
  - Doba odezvy aplikace
  - Chybové kódy a jejich frekvence
4. Databázový výkon:
  - Počet dotazů za sekundu
  - Využití připojení
  - Doba provádění dotazů
  - Využití tabulek a indexů

Každý dashboard je navržen tak, aby poskytoval relevantní informace pro různé role v týmu - od vývojářů po administrátory. Dashboards obsahují jak detailní grafy pro hlubokou analýzu, tak přehledové panely pro rychlé zjištění stavu systému.

### 5.3.6 Konfigurace Kuma

Uptime Kuma doplňuje náš monitorovací systém o sledování dostupnosti aplikace a jejích kritických komponent. Je přístupná na portu 3001 a vyžaduje počáteční nastavení po nasazení. Při prvním přihlášení vás Kuma vyzve k vytvoření admin účtu. Poté lze přidat monitorování pro různé endpointy aplikace.

### 5.3.7 Monitorované endpointy

Pro kompletní monitoring rezervačního systému monitorujeme následující endpointy:

1. Hlavní webová aplikace:
  - URL: <https://rezervace.spseplzen.cz>
  - Interval kontroly: 1 minuta
  - Metoda: GET
  - Očekávaný status kód: 200
2. Monitorovací služby:
  - Prometheus: <http://localhost:9090>
  - Grafana: <http://localhost:3000>
  - Kuma samotná: <http://localhost:3001>

Tento přístup zajišťuje, že jsme okamžitě informováni o jakýchkoli výpadech nebo degradaci služby, což umožňuje rychle reagovat na problémy.

## 5.4 Zálohování

### 5.4.1 Zálohování systému

Klíčovou součástí provozu aplikace jsou zálohy, které minimalizují riziko ztráty dat a umožňují rychlou obnovu v případě selhání. Pro zajištění maximální bezpečnosti dat jsme zajistili automatizované zálohy s definovaným intervalem a různými úložnými místy.



### 5.4.2 Strategie zálohování

Pro rezervační systém byla zvolena následující zálohovací strategie:

- Pravidelné automatické zálohy každé dva týdny
- Redundantní ukládání záloh - lokální zálohy na serveru a zálohy v cloudovém úložišti
- Rotace záloh - udržování definovaného počtu předchozích záloh s automatickým odstraňováním starších verzí
- Zálohování kódu i dat - kompletní záloha zahrnující jak zdrojový kód, tak databázový dump

### 5.4.3 Implementace zálohovacího systému

Zálohovací systém je implementován formou skriptů spouštěných plánovačem úloh cron:

1. Hlavní zálohovací skript - zajišťuje export databáze a vytvoření komprimovaného archivu celé aplikace
2. Skript pro cloudové zálohy - automaticky nahrává zálohy do cloudového úložiště MEGA

Zálohovací skripty jsou umístěny v adresáři /home/student/skripty/ a jsou nakonfigurovány s ohledem na specifické požadavky aplikace.

### 5.4.4 Obsah záloh

Každá záloha obsahuje všechny nezbytné komponenty pro úplnou obnovu systému:

- Zdrojový kód aplikace - všechny PHP, JavaScript a konfigurační soubory
- Databázový dump - kompletní export MySQL databáze s daty o rezervacích, uživateli a vybavení
- Konfigurační soubory - včetně Docker konfigurace a nastavení webového serveru
- Uživatelská data - nahrané soubory a média

Pro optimalizaci velikosti záloh jsou vyloučeny následující komponenty:

- Adresáře node\_modules a vendor (lze regenerovat pomocí npm a composer)
- Cache soubory a dočasné soubory (storage/framework/cache, storage/framework/sessions, storage/framework/views)
- Logy aplikace
- SSL certifikáty (lze obnovit)

### 5.4.5 Plánování záloh

Zálohy jsou automaticky spouštěny pomocí cron démona podle následujícího harmonogramu:

- Spuštění hlavního zálohovacího skriptu každou sudou neděli ve 2:00 ráno

- Spuštění skriptu pro cloud zálohy každou sudou neděli ve 3:00 ráno (hodinu po dokončení lokální zálohy)

#### **5.4.6 Úložiště záloh**

V souladu s osvědčenými postupy je implementována strategie více úložišť:

1. Lokální zálohy - uloženy v adresáři /home/student/zalohy na serveru
2. Cloudové zálohy - nahrávány do služby MEGA s šifrovaným přenosem a uložením

Tento přístup kombinuje výhody rychlého přístupu k lokálním zálohám s bezpečností offsite záloh v cloudu.

#### **5.4.7 Rotace a správa záloh**

Pro efektivní správu úložného prostoru je implementována automatická rotace záloh:

- Na lokálním úložišti jsou uchovávány 3 nejnovější zálohy
- V cloudovém úložišti je uchováno 6 nejnovějších záloh (přibližně 3 měsíce historie)
- Starší zálohy jsou automaticky odstraňovány

#### **5.4.8 Bezpečnostní aspekty**

Zálohovací systém je navržen s ohledem na bezpečnost dat:

- Komprese záloh pro minimalizaci velikosti
- Odstraňování citlivých informací ze záloh
- Omezený přístup k zálohovacím souborům (nastavení odpovídajících oprávnění)
- Zabezpečený přenos do cloudového úložiště

#### **5.4.9 Monitorování zálohovacího procesu**

Zálohovací skripty generují detailní logy o svém průběhu, které jsou uloženy v adresáři /home/student/skripty/logs/. Tyto logy obsahují informace o:

- Času zahájení a dokončení zálohy
- Úspěšnosti jednotlivých kroků
- Velikosti vytvořené zálohy
- Případných chybách a varováních

Tento přístup umožňuje snadnou kontrolu funkčnosti zálohovacího systému a rychlou diagnostiku případných problémů.

Implementovaný zálohovací systém poskytuje robustní řešení pro ochranu dat rezervačního systému a je důležitou součástí celkové infrastruktury zajišťující vysokou dostupnost a bezpečnost aplikace.

## 6 Co se nepovedlo

V současnosti není možné uvést projekt do provozu kvůli nefunkční autentizaci pomocí školního LDAP serveru. Nedá se však říct, že autentizace je nefunkční.

Ve verzi aplikace určené pro vývoj se automaticky vytváří lokální OpenLDAP server, který umožňuje aplikaci vyvíjet mimo virtuální server nacházející se ve školní síti. Tento server přesně kopíruje strukturu toho školního, aby nebylo nutné provádět žádné zásahy do kódu při nasazení do produkce. V této verzi web funguje bezchybně dle našich představ.

Problém nastává v produkční verzi, která se napojuje na školní LDAP server. Stejně jako v případě SQL relačních databází je pro napojení na LDAP server při využití knihovny LdapRecord nutné mít uživatele s oprávněním z databáze číst pro získání údajů jako je heslo, třída, jméno, email apod.

Z bezpečnostních důvodů nám však nebyl poskytnut účet, přes který bychom mohli k uživatelským údajům přistupovat, což kompletně zrušilo veškerou funkcionalitu aplikace v produkčním režimu. Tuto informaci jsme se dozvěděli týden před odevzdáním práce.

Dlouhou dobu jsme žili v iluzi, že přihlašování přes školní účty funguje, jelikož pro připojení k LDAP serveru jsme využívali vždy přihlašovací údaje jednoho z členů týmu, což fungovalo, jelikož sám k sobě uživatel práva pro čtení má.

Ve stejný den jsme se dozvěděli i to, že škola za pár měsíců přechází na jiný autentizační systém, který by tak i tak uvedl web do nefunkčního stavu, jelikož řešení pro stávající systém není kompatibilní s tím novým.

Velmi nás to všechny mrzí, jelikož projektu jsme věnovali velké množství času a nefunkčnost není ani naší vinou. Problematice týkající se autentizace uživatelů jsme se věnovali přinejmenším 2 poslední měsíce o obětovali řadu funkcí systému jen pro to, abychom zjistili, že autentizace webu přes veškeré naše snahy nebude fungovat.

### 6.1 Návrh řešení

Tento návrh počítá s použitím budoucího API pro přihlašování, o kterém jsme byli informováni příliš pozdě. Tak i tak bude nutný zásah do řady souborů týkající se autentizace i docker buildu. Bude zároveň nutné udělat rozhodnutí, zda ukládat uživatele do databáze nebo session. Vzhledem k četnému využití databázového modelu User napříč aplikací bychom však doporučili přihlášené uživatele ukládat do databáze pro snazší integraci s existujícím kódem a řídit se již existující databázovou strukturou.

Předpokládá se, že externí autentizační API bude po úspěšném ověření uživatele vracet přístupový token a základní informace o uživateli, jako je jeho unikátní identifikátor v externím systému, role (např. učitel, žák), třída, emailová adresa a jméno. Jelikož každý úspěšně přihlášený uživatel bude uložen nebo aktualizován v lokální databázové tabulce users, není potřeba implementovat další volání API pro získávání seznamu uživatelů nebo jejich detailů po přihlášení. Systém bude pracovat pouze s uživateli, kteří se aktivně přihlásili přes toto API a jsou tedy již zaznamenáni v lokální databázi.

Nejprve je nutné odstranit veškeré závislosti na knihovně LdapRecord a související infrastruktuře. To zahrnuje odinstalování balíčku directorytree/ldaprecord-laravel pomocí Composeru, smazání konfiguračního souboru config/ldap.php a odstranění případných souvisejících proměnných prostředí z .env.local a .env.production.example. Dále je třeba smazat adresář app/Ldap. V rámci úklidu je také vhodné odstranit definici služby OpenLDAP z docker-compose.yml a smazat příslušné Docker image a volumes.

Následně je potřeba upravit model App\Models\User. Odstraňte jakékoli dědění z LDAP modelů nebo použití LDAP traitů. Ujistěte se, že model dědí ze standardní třídy Illuminate\Foundation\Auth\User as Authenticatable. Přidejte do modelu a odpovídající databázové migrace nové sloupce pro ukládání informací získaných z API. Sloupec password v lokální databázi pravděpodobně nebude pro přihlášení využíván. Zajistěte, aby nové sloupce byly přidány do \$fillable pole modelu pro umožnění jejich hromadného přiřazení.

Dalším krokem je úprava konfigurace autentizace v souboru config/auth.php. Změňte výchozí guard v sekci defaults na nový, vlastní guard (např. api\_token). V sekci guards definujte tento nový api\_token guard a specifikujte, že bude používat vlastní driver. V sekci providers definujte nebo upravte providera pro uživatele (např. users), který bude používat eloquent driver a odkazovat na upravený model App\Models\User, jelikož uživatelé budou spravováni lokálně po přihlášení.

Klíčovou změnou je implementace vlastní autentizační logiky. Vytvořte nový nebo použijte existující AuthServiceProvider. V jeho boot metodě použijte Auth::extend() pro registraci vlastního driveru (např. api\_token\_driver). Tento driver bude obsahovat logiku pro komunikaci s externím API. Při pokusu o přihlášení (Auth::attempt()) tento driver přijme přihlašovací údaje (např. email a heslo), odešle je na externí autentizační API, zpracuje odpověď, a v případě úspěchu získá token a data uživatele. Následně vyhledá nebo vytvoří (updateOrCreate) záznam uživatele v lokální databázi users pomocí dat z API (zejména external\_id a email). Po úspěšném nalezení nebo vytvoření lokálního uživatele vrátí jeho instanci Laravelu, čímž dojde k přihlášení uživatele do aplikace (typicky pomocí session). V případě neúspěšného ověření přes API vrátí null. Metoda pro odhlášení v rámci guardu by měla zajistit lokální odhlášení

(zrušení session) a případně zavolat endpoint pro zneplatnění tokenu na straně externího API, pokud to API vyžaduje.

Nakonec upravte AuthController. V metodě pro přihlášení nahraďte volání související s LDAP za volání `Auth::guard('api_token')->attempt($credentials)`. Zpracujte výsledek tohoto volání pro přesměrování uživatele nebo zobrazení chybové hlášky. V metodě pro odhlášení použijte `Auth::guard('api_token')->logout()` a přidejte případné volání externího API pro odhlášení. Ujistěte se, že přihlašovací formulář stále sbírá potřebné údaje pro externí API. Po implementaci všech změn je nezbytné důkladně otestovat celý proces přihlášení, registraci/aktualizaci uživatele v lokální DB, odhlášení a ověření přístupových práv na základě rolí uložených u lokálního uživatele.

## Závěr

V průběhu práce na tomto projektu jsme se seznámili s celou řadou moderních technologií a postupů, které byly pro nás zpočátku neznámé. Bohužel jsme nebyli schopni přes veškerou snahu zprovoznit autentizaci pomocí školních účtů, což je klíčová část aplikace.

Během vývoje jsme si osvojili práci s frameworkem Laravel a jeho propojením s JavaScriptovou knihovnou React prostřednictvím Inertia.js, což nám umožnilo vytvořit dynamickou a intuitivní aplikaci, která disponuje moderním uživatelským rozhraním kompatibilním s různými velikostmi obrazovek. Povedlo se nám zautomatizovat řadu procesů a optimalizovat jejich zpracování pro dosažení nejlepšího uživatelského zážitku.

Dále jsme se zabývali problematikou zabezpečení, nasazení aplikace na server pomocí Dockeru a optimalizací výkonu. Implementovali jsme notifikační systém pomocí emailů a uživatelské autentizace prostřednictvím LDAP serveru, který kopíruje strukturu toho školního.

Přestože výsledkem naší práce není zatím využitelný systém, věříme, že jsme udělali vše pro jeho zprovoznění a popsali postup, kterým by bylo možné aplikaci v budoucnu uvést do provozu. Současně lze projekt spustit ve verzi pro lokální vývoj a slouží pouze jako technické demo. Věříme, že v případě zprovoznění autentizace by se jednalo o velmi funkční řešení, které by zavedlo řád a efektivitu do rezervací školního vybavení.

## Literatura

1. Pages. *Inertia.js*. [Online] n.d. [Citace: 10. březen 2025.] <https://inertiajs.com/pages>.
2. The protocol. *Inertia.js*. [Online] n.d. [Citace: 10. březen 2025.] <https://inertiajs.com/the-protocol>.
3. How it works. *Inertia.js*. [Online] n.d. [Citace: 10. březen 2025.] <https://inertiajs.com/how-it-works>.
4. Links. *Inertia.js*. [Online] n.d. [Citace: 10. březen 2025.] <https://inertiajs.com/links>.
5. Forms. *Inertia.js*. [Online] n.d. [Citace: 10. březen 2025.] <https://inertiajs.com/forms>.
6. Shared data. *Inertia.js*. [Online] n.d. [Citace: 11. březen 2025.]
7. React Reference Overview. *React*. [Online] n.d. [Citace: 9. březen 2025.] <https://react.dev/reference/react>.
8. Usage. *LdapRecord*. [Online] 8. listopad 2024. [Citace: 10. březen 2025.] <https://ldaprecord.com/docs/laravel/v3/usage#models>.
9. Introduction. *React Day Picker*. [Online] n.d. [Citace: 11. březen 2025.] <https://daypicker.dev/>.
10. Routing. *Laravel*. [Online] n.d. [Citace: 9. březen 2025.] <https://laravel.com/docs/11.x/routing>.
11. Authorization. *Laravel*. [Online] n.d. [Citace: 10. březen 2025.] <https://laravel.com/docs/11.x/authorization>.
12. Middleware. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/middleware>.
13. Validation. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/validation>.
14. CSRF Protection. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/csrf>.
15. Database Transactions. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/database#database-transactions>.



16. Eloquent: Getting Started. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/eloquent>.
17. Eloquent: Relationships. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/eloquent-relationships>.
18. Soft Deleting. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/eloquent#soft-deleting>.
19. Task Scheduling. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/scheduling>.
20. Notifications. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/notifications>.
21. Flash Data. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/session#flash-data>.
22. Queues. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/queues>.
23. Testing: Getting Started. *Laravel*. [Online] n.d. [Citace: 11. březen 2025.] <https://laravel.com/docs/11.x/testing>.
24. Authentication. *Laravel*. [Online] n.d. [Citace: 10. březen 2025.] <https://laravel.com/docs/11.x/authentication>.

## Seznam obrázků

|  |    |
|--|----|
| Obrázek 1: ER diagram .....  | 18 |
| Obrázek 2: návrh přehledu rezervací žáka .....                             | 20 |
| Obrázek 3: návrh katalogu dostupného vybavení .....                        | 20 |
| Obrázek 4: návrh detailu vybavení z katalogu .....                         | 21 |
| Obrázek 5: návrh administrátorského dashboardu .....                       | 21 |
| Obrázek 6: návrh schvalovací stránky rezervace .....                       | 22 |
| Obrázek 7: přihlašovací formulář .....                                     | 23 |
| Obrázek 8: přehled probíhajících rezervací .....                           | 24 |
| Obrázek 9: katalog vybavení .....  | 24 |
| Obrázek 10: detailní stránka s otevřeným kalendářem .....                  | 25 |
| Obrázek 11: detailní stránka v tmavém režimu .....                         | 25 |
| Obrázek 12: administrátorský dashboard .....                               | 26 |
| Obrázek 13: výpis neschválených žádostí .....                              | 27 |
| Obrázek 14: stránka pro schválení žádosti .....                            | 27 |
| Obrázek 15: stránka Vybavení .....   | 28 |
| Obrázek 16: detail kategorie .....   | 29 |
| Obrázek 17: příklad nastavení jiného než defaultního layoutu .....         | 32 |
| Obrázek 18: příklad zpřístupnění předávaných dat .....                     | 34 |
| Obrázek 19: příklad přesměrování s předáním hodnot .....                   | 35 |
| Obrázek 20: příklad výpisu pole hodnot .....                               | 36 |
| Obrázek 21: příklad použití podmíněných stylů .....                        | 36 |
| Obrázek 22: příklad policy zabráňující změně vybavení cizích správců ..... | 42 |

|   |    |
|---|----|
| Obrázek 23: cron úloha pro mazání archivovaných rezervací ..... | 45 |
| Obrázek 24: přidání obrázku do fronty.....                      | 49 |