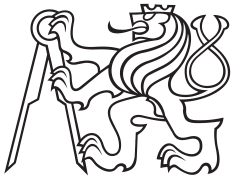
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Návrh a implementace kolaborativní kreslící aplikace - hry

Iryna Netreba

Vedoucí: RNDr. Ladislav Serédi
Studijní program: Softwarové inženýrství a technologie
Květen 2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Netreba** Jméno: **Iryna** Osobní číslo: **494747**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

Studijní obor: **bez oborů**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Návrh a implementace kolaborativní kreslící aplikace - hry

Název bakalářské práce anglicky:

Design and implementation of a collaborative painting application – game

Pokyny pro vypracování:

Proveďte rešerši trhu kolaborativních kreslících aplikací a porovnejte existující řešení z pohledu uživatele i z hlediska použitých technologií. Identifikujte jejich výhody a nevýhody.
Z dostupné literatury prostudujte teoretické základy problematiky kolaborativních kreslících aplikací.
Navrhněte architekturu webové aplikace založené na moderních technologiích – jako například jazyk Dart a framework Flutter - implementující výše popsaný systém. Analyzujte vhodnost frameworku Flutter a porovnejte ho s jinými technologiemi z hlediska požadavků aplikace. Back-end, tedy server, implementujte ve framework Spring Boot. Komunikace s front-end bude probíhat pomocí REST API.
Aplikace bude zajišťovat následující funkcionalitu:
• Kreslící hra: Hlavním cílem této části aplikace bude umožnit uživatelům nakreslit obrázek podle zvolené šablony.
• Režim tvorby šablon: Uživatelé budou moci vytvářet vlastní šablony, ukládat je a sdílet s ostatními uživateli.
• Knihovna kreslících šablon: V této části budou uživatelé schopni stahovat různé šablony z knihovny a kreslit podle nich.
Na základě vašeho návrhu implementujte funkční kreslící aplikaci tak aby bylo možné otestovat klíčové uživatelské scénáře. Výsledky testů vyhodnoťte, diskutujte dosažené výsledky, případně i zjištěné nedostatky a možný budoucí vývoj aplikace.

Seznam doporučené literatury:

Kooperativní mobilní multiplatformní hra, Jan Bittner, bakalářská práce FEL ČVUT on-line: https://dspace.cvut.cz/handle/10467/88709
Collaborative VR painting in web browsers, Jonathan Knispel, Fraser Bullock, SIGGRAPH Asia 2017 on-line: https://dl.acm.org/doi/abs/10.1145/3139468.3148451
Designing a Collaborative Finger Painting Application for Children, Ben Bederson, Allison Druin, Lisa Sherman, October 2000, on-line: https://www.researchgate.net/publication/2815298_Designing_a_Collaborative_Finger_Painting_Application_for_Children
ArtWorkout: Learn How to Draw, on-line: https://apps.apple.com/us/app/artworkout-learn-how-to-draw/id1564657118
Aggie.io A collaborative painting application by Magma on-line: https://aggie.io/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Ladislav Serédi kabinet výuky informatiky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **02.02.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

RNDr. Ladislav Serédi podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

.	
Datum převzetí zadání	Podpis studentky

Poděkování

Chtěla bych poděkovat vedoucímu své práce RNDr. Ladislavu Serédi za podporu, rady, pěkné vedení této práce a motivaci poznávat nové pro mě technologie. Ráda bych také poděkovala své rodině za podporu a trpělivost během mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracovala samostatně, a že jsem uvedla veškerou použitou literaturu.

V Praze, 24. května 2024

Abstrakt

Tato bakalářská práce se zabývá analýzou, návrhem a implementací multiplatformní kolaborativní kreslící aplikace, která umožňuje uživatelům překreslovat obrázky podle šablon. Uživatelé mohou najít vlastní účel použití aplikace: zábava, sdílení kreslících schopností a vlastní tvorby, interaktivní kreslící cvičení, výzva osobní přesnosti překreslování.

Práce obsahuje porovnání existujících řešení, analýzu použitých technologií, definované požadavky, uživatelské scénáře, a návrh aplikace.

Praktická část práce obsahuje implementaci, kde jsou popsány stěžejné funkce a struktura aplikace. Dále následuje popis provedeného nasazení aplikace a uživatelské testování.

Klíčová slova: Flutter, Kreslení, Šablona, Multiplatformní aplikace, Spring Boot

Vedoucí: RNDr. Ladislav Serédi

Abstract

This bachelor thesis deals with the analysis, design, and implementation of a multiplatform collaborative drawing application that allows users to redraw images according to templates. Users can find their own purpose for using the application: entertainment, sharing drawing skills and creations, interactive drawing exercises, personal challenge of redrawing accuracy.

The thesis includes a comparison of existing solutions, an analysis of the technologies used, defined requirements, user scenarios, and application design.

The practical part of the thesis includes implementation, describing characteristic features and application structure. It is followed by a description of the application deployment and user testing.

Keywords: Flutter, Drawing, Template, Multiplatform application, Spring Boot

Title translation: Design and implementation of a collaborative painting application – game

Obsah

Obrázky

Uvod

Každým rokem se digitální kreslení více popularizuje. Pro tento způsob kreslení stačí pouze mobilní telefon nebo počítač. S rostoucím zájmem o digitální kreslení se zvyšuje poptávka po aplikacích, které nabízejí různé možnosti grafického vyjádření. Nikoho již nepřekvapí kreslící aplikace typu Paint, ta je pro současnost příliš jednoduchá. Uživatelé hledají zajímavá řešení, která mohou nabídnout neobvyklé zážitky: společné kreslení v online režimu [1], kreslení ve virtuální realitě [2], aplikace, které v interaktivní formě učí lidi kreslit [3]. Dokonce existuje celá řada různých zařízení, jejichž cílem je přenést proces kreslení do digitální podoby, například grafické tablety, zařízení s multi-touch obrazovkou [4] a podobně.

Nicméně nemají všichni lidé přirozený talent kreslit, avšak stále mohou mít zájem o tvorbu obrázků vlastníma rukama. Existuje mnoho různých cvičení, která pomáhají lidem se naučit kreslit, ale většina dostupných online zdrojů je spíše teoretická, protože vytváření interaktivních online cvičení je docela náročné. Vzhledem k těmto důvodům je zajímavou zkušenosti vytvořit aplikace, ve které lidé budou schopni využívat předpřipravené kreslící šablony, podle kterých se dá překreslit obrázky pro zábavu. Předpokládá se, že šablony vytvářejí další uživatelé a nahrávají je do veřejné knihovny. Z knihovny je potom možné šablony stáhnout a překreslit je.

Tato práce obsahuje analýzu, návrh, implementaci a testování kolaborativní kreslící aplikace. Cílovou skupinou aplikace jsou převážně děti a mládež.

Kapitola 1

Cíl práce

Cílem práce je analýza, návrh, implementace a testování kolaborativní kreslící aplikace, ve které uživatelé mají možnost překreslit obrázky podle šablon a vytvářet vlastní šablony pro další uživatele.

V rámci teoretické části práce je provedena analýza a zhodnocení existujících řešení, identifikace jejich výhod a nevýhod pro vhodnou inspiraci. Dále následuje analýza moderních technologií, porovnání a výběr nástrojů s ohledem na potřeby aplikace. Dalším důležitým úkolem je navržení požadavků, architektury, logiky aplikace a definování uživatelských scénářů.

V implementační části budou ukázané naprogramované klíčové detaily aplikace, jako například vlastní kreslení na klientské straně a ukládání šablon na serveru. Dále bude následovat popis nasazení aplikace. V kapitole Testování bude popsáno uživatelské testování podle definovaných scénářů.

Kapitola 2
Analýza

2.1 Analýza existujících řešení

V následujících sekcích naleznete analýzu stávajících řešení. Po prozkoumání aplikací podobného zaměření (kreslící hry, velká knihovna kreslících šablon, kooperativní aplikace umožňující uživatelům poskytovat zpětnou vazbu na cizí kresby a využívat je pro hru) byly vybrány a analyzovány konkurenční kreslící aplikace. Výběr těchto aplikací byl založen na jejich popularitě a podobnosti s systémem navrženým v rámci této práce.

2.1.1 ArtWorkout

ArtWorkout [3] je kreslící aplikace určená pro iOS. Uživatel si vybírá jeden obrázek ze široké knihovny šablon a spouští takzvanou kreslící hru, jejímž cílem je co nejvěrněji nakreslit zvolený obrazek. Kreslení je rozděleno do kroků, kde každý krok jasně ukazuje uživateli, co má nakreslit a jak, takže slouží jako šablona. Každý krok je ohodnocen na základě přesnosti provedení, a na konci hry uživatel obdrží body za přesnost.

Celkově lze tuto aplikaci charakterizovat jako pokročilou verzi omalovánky, která má za cíl naučit uživatele kreslit.

Klady:

- Rozsáhlá knihovna omalovánek.
- Jednoduché a intuitivní uživatelské rozhraní.
- Citlivost na tlak dotykového pera.

Zápory:

- Více než 85% obsahu je za poplatek.
- Není možné vytvářet vlastní kreslící šablony.
- Aplikace je omezena lokálním použitím na zařízení, nelze se spojit s jinými hráči a sdílet například počet bodů nebo výsledný obrázek po dokončení kresby.
- Dostupná pouze na platformě iOS.

2.1.2 Aggie.io

Aggie.io [1] je kreslící aplikace určená pro webové prohlížeče. Hlavním cílem této aplikace je umožnit kreslení pro více uživatelů na jednom plátně současně. Každý uživatel vidí v reálném čase změny provedené jinými uživateli na plátně. Pro sdílení přístupu k plátnu musí hostitel platna poslat odkaz ostatním uživatelům. Aplikace disponuje velmi podrobnými možnostmi nastavení plátna, tužky a dalších kreslících atributů. Během kreslení je také možné chatovat a volat přímo v rámci této aplikace. K dispozici je také správa uživatelů, kde každý uživatel má svou stránku s seznamem všech pláten, ke kterým má přístup.

Klady:

- Kolektivní kreslící režim, majitel plátna může pozvat kohokoliv prostřednictvím odkazu na plátno.
- Rozsáhlý výběr nastavení pro všechny kreslící nástroje, plnohodnotná kreslící aplikace.

Zápory:

- Offline režim pro jednotlivce není podporován. Aplikace vyžaduje přístup k internetu, protože funguje v reálném čase.
- Uživatel má přístup k cizím plátnům pouze v případě, když obdrží pozvání od majitele plátna; veřejný přístup, a to ani k funkcím jako komentování, není podporován.

2.1.3 Coloring Book for Me [5], Colorfy: Colouring Book Games, Adult Colouring Book - Pigment a další podobné aplikace

Jedná se o omalovánky, které jsou dostupné na iOS, a jsou si velmi podobné, takže analýza pro ně platí obecně. Tyto aplikace nabízejí knihovnu šablon, které lze vybarvit. Uživatel nemusí vytvářet vlastní obrázek, protože šablona je předem definovaná a uživatel ji pouze vybarvuje. Po vybarvení lze obrázek stáhnout nebo sdílet na sociálních sítích.

Klady:

- Široká škála nástrojů pro malování, včetně tužek, kartáčů a dalších.
- Citlivost na tlak dotykového pera.

Zápory:

- Téměř všechny funkce a šablony v knihovně jsou placené.
- Aplikace funguje lokálně, nelze se spojit s jinými uživateli.

2.1.4 Vyhodnocení

Aplikace, která je navržena v rámci této práce, je inspirována některými částmi výše zmíněných programů. Tyto programy nejsou zcela totožné s hlavním konceptem této práce, ale obsahují některé zajímavé prvky. První aplikace přináší zajímavý proces kreslení podle šablon. Druhá aplikace poskytuje inspiraci pro správu kreslících pláten. Třetí příklad se zaměřuje na základní myšlenku vybarvování obrázků podle šablon a také možnost sdílení výsledných obrázků. Z uvedených výhod a nevýhod těchto aplikací lze čerpat inspiraci a formulovat požadavky pro nové řešení.

2.2 Analýza technologií

2.2.1 Frontend

Základním krokem návrhu systému je zaměření na specifickou funkcionalitu programu, na základě níž lze odvodit seznam platforem, které poskytnou nejlepší uživatelskou zkušenost při používání aplikace. Jedním ze zaměření této práce je vhodný způsob kreslení, a v současné době existují různé platformy, které tuto možnost nabízejí různými způsoby. Mnoho tabletů je vybaveno dotykovými perami. Platformy jako macOS, Windows, Linux a další podporují grafické tablety s důležitou vlastností, jako je citlivost na sílu tlaku. Mobilní senzorická zařízení umožňují kreslení pomocí dotyku prstů.

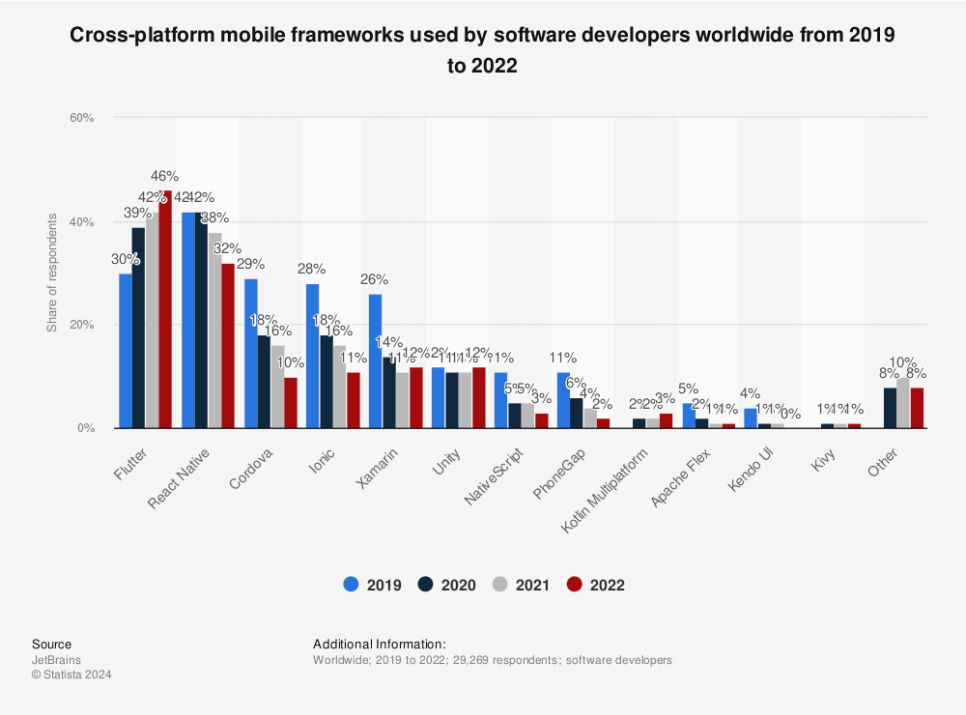
Jednotlivé knihovny, dostupné pro různé programovací jazyky, podporují kreslení do různé míry. V rámci této práce bylo zvažováno použití Javy a vytvoření desktopové aplikace pro kreslení pomocí grafických tabletů. Klíčovou funkcí této aplikace měla být podpora tlakově citlivého pera. Prvním vyzkoušeným nástrojem pro realizaci této myšlenky byla knihovna JPen ve verzi 2-150301 [6], která tuto funkcionalitu nabízí. Byla zvolena proto, že během rešerše technologií a existujících experimentů byla nalezena práce [7], kde tato knihovna byla použita pro kreslící aplikace. Problémem je to, že i když knihovna poskytuje podporu citlivosti pera, JPen je docela zastaralá technologie a používá Java 5. Poslední verze knihovny pochází z roku 2015, a proto byla tato varianta zamítnuta.

Následně bylo uvažováno o vytvoření webové aplikace s využitím frameworku React. Situace ohledně aktuálnosti technologií je lepší, existuje i TypeScript knihovna Perfect-Freehand [8], která poskytuje podporu tlakově citlivého pera. To znamená, že tato aplikace by mohla být plně využívána v prohlížeči pomocí grafického tabletu na stolním počítači a na dotykových tabletech s perem, například na iPad, Samsung Galaxy Tab a dalších podobných zařízeních. Samozřejmě by aplikace byla přístupná i prostřednictvím jakéhokoliv zařízení s přístupem k internetu a prohlížečem, ale ne každé dotykové zařízení podporuje citlivost na tlak prstu. Nevýhodou tohoto řešení je, že taková aplikace vyžaduje trvalé připojení k internetu, jinak by nebylo možné kreslení uložit na server. Použití webových aplikací v mobilním prohlížeči na dotykovém zařízení take není moc užitečné.

Další zajímavou variantou bylo použití frameworku Electron [9] ve spojeni s React. Electron umožňuje vývoj multiplatformních desktopových aplikací. Framework poskytuje přístup k nativním prvkům operačních systémů, například k souborovému systému. Problém tohoto řešení spočívá v tom, že Electron může být náročný na výkon a paměť, protože spouští vlastní instance webového prohlížeče pro každou aplikaci. Instalační balíčky aplikací vytvořených s Electronem mají tendenci být větší kvůli balení webového prohlížeče a jiných závislostí.

Vzhledem k výše uvedeným důvodům, zajímavým řešením bude navrhnout multiplatformní aplikaci, která optimalizuje proces vývoje takové aplikace. Výhodou tohoto řešení je, že zdrojový kód je jednotný pro všechny platformy, což zjednoduší vývoj, podporu a aktualizace aplikace.

Porovnani nastroju



Obrázek 2.1: Multiplatformní frameworky, ktere pouzivaji vyvojare po celém světě z 2019 po 2022 [10]

Tento obrázek ukazuje graf frekvence používání různých frameworků pro vývoj multiplatformních aplikací. Z grafu je vidět, že v současné době největší popularitu má framework Flutter [11], a s každým rokem roste procento jeho používání. Na druhém místě je React Native [12], u kterého naopak s průběhem času popularita klesá. To jsou frameworky, které se pro multiplatformní vývoj nejčastěji využívají. Ostatní frameworky v roce 2022 mají dvakrát až třikrát menší zastoupení. V rámci této práce bude provedeno porovnání dvou nejpopulárnějších frameworků: React Native a Flutter.

React Native

React Native je framework pro vývoj multiplatformních mobilních aplikací. Jedná se o framework postavený na jazyce JavaScript, který umožňuje vývojářům psát mobilní aplikace, které mohou běžet na více platformách, včetně iOS a Android. Vývojáři píšou většinu kódu v jazyce JavaScript nebo TypeScript. Tento kód může obsahovat veškerou logiku aplikace, uživatelské rozhraní a interakci. Jedna z klíčových vlastností React Native je schopnost převést komponenty napsané v JavaScriptu na nativní komponenty pro konkrétní platformu. Tímto způsobem může aplikace vytvářet nativní uživatelské rozhraní, které vypadá a chová se jako aplikace napsaná přímo pro danou platformu. Nutnost komunikovat s nativním kódem může vést k menšímu výkonu. Další výhodou tohoto frameworku je velká komunita a rozsáhlá knihovna třetích stran. Co se tyče distribuce, balíčky React Native aplikace jsou obvykle menší, protože většina logiky je v nativním kódu.

```
<View>
     <Button onPress = {} title = "Hello␣world"/>
</View>
```

Listing 2.1: Příklad komponenty tlačitka

Flutter

Flutter je open-source framework vyvinutý společností Google pro vývoj multiplatformních mobilních a webových aplikací. Jedná se o framework postavený na programovacím jazyku Dart. Flutter se vyznačuje tím, že umožňuje vytvářet uživatelsky přívětivé a rychlé aplikace s jediným kódovým základem, který může běžet na různých platformách, včetně iOS, Androidu, webu a desktopu. Flutter používá programovací jazyk Dart [13], který byl vyvinut také společností Google. Widgety jsou základními stavebními bloky rozhraní aplikace. Flutter umožňuje jak používat vestavěné widgety tak i vytvářet vlastní widgety a kombinovat je do složitých rozhraní. Flutter má vlastní grafický engine pro vykreslování uživatelského rozhraní, což umožňuje rychlé a plynulé vykreslování. Framework umožňuje vytvářet aplikace s konzistentním designem na všech platformách. Ale protoze jazyk Dart není tak populární, počet dostupných knihoven a komunita jsou menší v porovnani s jazyky JavaScript nebo TypeScript. Flutter často nabízí lepší výkon díky vlastnímu renderingovému enginu a kompilovanému kódu. Ale na rozdíl od Reactu, Flutter aplikace jsou často větší, protože obsahují knihovny Dart. To může ovlivnit velikost balíčků pro stažení.

```
ElevatedButton(
  onPressed: () {},
  child: Text('Hello␣world'),
)
```

Listing 2.2: Příklad widgetu tlačítka

Programovací jazyk Dart

Dart je moderní programovací jazyk vyvinutý firmou Google. Byl navržen tak, aby byl výkonný, efektivní a přátelský k vývoji webových a mobilních aplikací. Dart má jednoduchou a srozumitelnou syntaxi, což dělá ho snadno čitelným a pohodlným jazykem. Je podobný některým jiným jazykům, jako jsou Kotlin, JavaScript, C#, což usnadňuje učení a přechod pro vývojáře, kteří znají tyto jazyky. Kód napsaný v Dartu je přeložen do nativního kódu pro konkrétní platformy, což zvyšuje jeho výkon. Dart podporuje asynchronní programování:

- **Future**

  Základním prvkem asynchronního programování v Dartu jsou tzv. „futures“. Future reprezentuje hodnotu, která bude dostupná v budoucnu, ale nezabrání aplikaci v běhu. Když se spustí asynchronní operace, jako je stahování dat z internetu, aplikace získá Future, který bude dokončen, až když operace skončí. Během čekání na dokončení je možné provádět jiné úkoly.

- **Async a Await**

  Klíčová slova „async“ a „await“ jsou použita k označení asynchronního kódu. Pokud funkce je oznacena jako „async“, je mozne v ní použít „await“ k zastavení běhu funkce, dokud nejsou dokončeny asynchronní operace. To znamená, že je možné čekat na výsledek asynchronní operace, aniž by zbytek aplikace se zablokoval.

Dalším důležitým pojmem je widget. V Dart a frameworku Flutter jsou „widgety“ základním stavebním prvkem uživatelského rozhraní. Widgety jsou jednotky UI, které reprezentují různé části aplikace, od jednoduchých prvků, jako jsou tlačítka a textová pole, po složitější prvky, jako jsou seznamy, formuláře a obrazovky. Widgety jsou použity k tomu, aby se mohly sestavit a strukturovat uživatelské rozhraní.

- **Stavový widget (Stateful widget)**

  Stavový widget má schopnost udržovat a reagovat na svůj vlastní stav, který může být dynamicky změněn během životního cyklu widgetu. Stavové widgety jsou důležité pro vytváření interaktivních prvků uživatelského rozhraní, které mohou reagovat na uživatelské interakce, aktualizovat se a představovat různé data nebo stavy.

- **Stavově nezávislý widget (Stateless widget)**

  Stavově nezávislý widget nemá žádný vnitřní stav, který by se mohl měnit. Takovy widget je statický a nejedná se o dynamický prvek, který by mohl uchovávat data nebo reagovat na změny. Jedná se o čistě prezentační widget, který se používá k vykreslení statického obsahu nebo ke zobrazení dat, která se nemění v průběhu životního cyklu widgetu.

- **Inherited Widget**

  Inherited Widget vytváří strom sdílených dat, které jsou dostupné potomkům obaleným odpovídajícím Inherited Widgetem. Pokud se data v Inherited Widget změní, všechny potomky, kteří přistupují k těmto datům, budou automaticky aktualizovány. Inherited Widget je užitečný v situacích, kdy je potřeba sdílet stav nebo data mezi více widgety bez nutnosti předávání dat od widgetu k widgetu. To může výrazně zjednodušit a zpřehlednit kód, zejména v případě složitějších aplikací, kde existuje více widgetů, které potřebují přistupovat k společným datům.

```
class MojeAplikace extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Moje␣Aplikace'),
        ),
        body: Center(
          child: Text('Vítejte␣v␣mé␣aplikaci!'),
        ),
      ),
    );
  }
}
```

Listing 2.3: Ukázka jednoduchého widgetu

V tomto příkladu je vytvořen widget "MojeAplikace", který rozšiřuje "StatelessWidget". V rámci funkce "build()" je definováno uživatelské rozhraní pro aplikace. Tato aplikace obsahuje jednoduchý textový widget umístěný uprostřed obrazovky. Funkce "build()" je klíčovým bodem pro vytvoření a vrácení vizuálního stromu (Widget stromu), který reprezentuje uživatelské rozhraní aplikace.

2.2.2 Backend

Další nutnou a důležitou částí celkové aplikace představuje serverová strana, která je nezbytná pro účely daného projektu pro uchovávání uživatelských dat a poskytování byznys logiky. V dnešní době existuje dost populárních řešení pro implementaci serverové strany. V této práci bude provedeno porovnání několika z nich.

Django

Django [14] je framework napsaný v jazyce Python, který nabízí širokou škálu funkcí přímo „z krabice". Mezi tyto funkce patří:

- komunikace s databází pomocí Python objektů místo přímých SQL dotazů,
- systém pro autorizaci uživatelů
- automaticky generované, dynamicky přizpůsobitelné uživatelské rozhraní pro správu obsahu webové aplikace
- podpora pro vytváření middleware vrstev

Ačkoliv je Django škálovatelné, může vyžadovat více práce a optimalizací pro aplikace s extrémně vysokým výkonem.

Django je obzvláště oblíbené v situacích, kde jsou potřeba rychlé iterace a kde se hodí robustní databázové řešení s podporou většího množství uživatelů a bezpečnostních požadavků.

Laravel

Laravel [15] je moderní framework pro vývoj webových aplikací napsaný v PHP. Je vysoce oblíbený mezi vývojáři díky své eleganci, jednoduchosti a rozsáhlým funkcím, které zjednodušují vývoj aplikací. Laravel je známý svým expresivním, uživatelsky přívětivým syntaxem a je navržen s důrazem na dostupnost a flexibilitu. Mezi hlavní rysy Laravelu patří:

- objektově-relační mapování, které umožňuje pracovat s databázemi pomocí PHP objektů
- migrace databáze
- výkonný systém šablon, který umožňuje vkládat PHP kód přímo do HTML
- intuitivní a jednoduchý způsob definování tras a přidružení k nim kontrolerů

Z druhé strany, Laravel může trpět problémy s výkonem při škálování na velké aplikace [16], správné nastavení a optimalizace mohou tyto problémy řešit jen částečně. Kromě toho, velká závislost na balíčcích třetích stran může vést k problémům s údržbou a aktualizacemi.

Express.js

Express.js [17] je minimalistický a flexibilní webový aplikační framework pro Node.js, který poskytuje soubor robustních funkcí pro webové a mobilní aplikace. Je jedním z nejpopulárnějších frameworků pro vývoj serverové strany v JavaScriptu. Express zjednodušuje proces vývoje tím, že poskytuje nástroje pro správu trasy, zpracování požadavků a odpovědí a snadné používání middleware. Jako minimalistický framework nemá mnoho vestavěných bezpečnostních funkcí nebo konvenčních struktur, což může znamenat více práce pro vývojáře při zabezpečování aplikace.

Spring Boot

Spring Boot [18] je rozšíření populárního frameworku Spring, které usnadňuje vývoj nových Spring aplikací s minimální konfigurací. V Spring Boot mnoho aspektů aplikace je automaticky nakonfigurováno na základě obecně přijímaných pravidel, čímž se značně urychluje vývoj a zjednodušuje nasazení. Může být obtížné se naučit kvůli své komplexnosti a rozsáhlému ekosystému. Klíčové vlastnosti frameworku:

- Spring Boot automaticky nastaví aplikaci na základě deklarovaných závislostí
- Spring Boot aplikace lze spustit jako nezávislé Java aplikace, které obsahují vložený webový server bez nutnosti nasazení na externí server
- podpora pro různé typy aplikací, snadná integrace s dalšími Spring projekty, například Spring Data, Spring Security a další
- Spring Boot poskytuje aktuátory - sady přednastavených endpointů pro monitorování a správu aplikace ve výrobním prostředí.

Nicméně, aplikace vytvořené pomocí Spring Boot mohou být těžkopádné s pomalejším startem. Tato vlastnost může být problém pro některé typy nasazených služeb.

Zhodnocení

Pro potřeby daného projektu jsou vhodné všechny výše uvedené frameworky. Každý z nich svým způsobem podporuje REST API, poskytuje funkce nutné pro aplikace, jako například časté předávání mezi klientem a serverem multipart souborů. Proto dalším faktorem, který ovlivňuje výběr technologie, je zkušenost autora práce a jazyková preference. Na základě těchto důvodů byl pro serverovou část projektu vybrán framework Spring Boot.

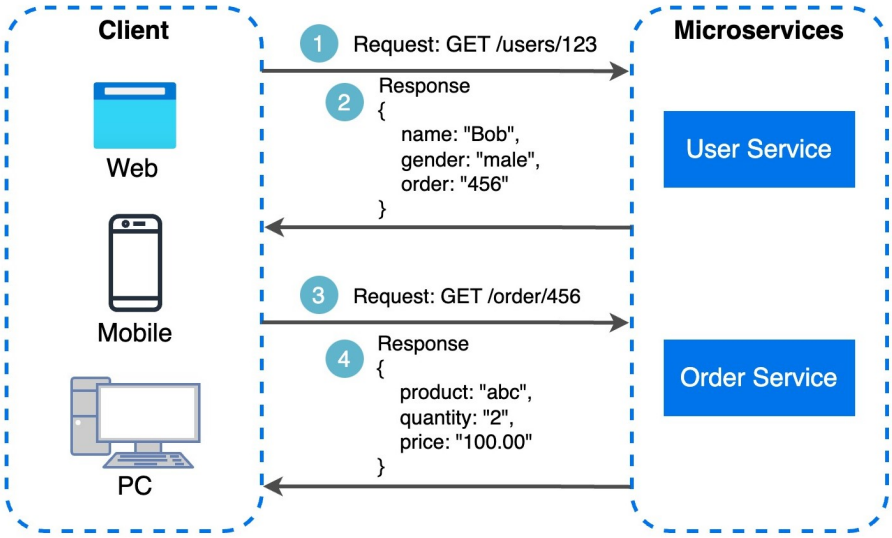
2.2.3 REST API

REST API je sada pravidel a konvencí pro návrh a komunikaci webových služeb. REST je architektonický styl, který klade důraz na jednoduchost a škálovatelnost webových služeb. RESTful API umožňuje komunikaci mezi různými softwarovými systémy přes HTTP, což je protokol používaný pro přenos dat na internetu.

Hlavními rysy REST API jsou:

- Zdroje: Všechny informace jsou vystaveny jako zdroje. Zdroje jsou identifikovány jedinečnými URI.
- Stav: Každý zdroj může mít různé stavy, jako je zobrazení, vytvoření nebo smazání.

- Operace: REST API definuje několik základních operací, které mohou být prováděny nad zdroji. Tyto operace jsou obvykle realizovány pomocí standardních HTTP metod, jako jsou GET (pro čtení dat), POST (pro vytváření nových dat), PUT (pro aktualizaci dat) a DELETE (pro odstranění dat).
- Bezstavovost: Každý požadavek na RESTful API by měl obsahovat všechny potřebné informace pro pochopení a vyřešení tohoto požadavku. Server by neměl ukládat informace o stavu klienta mezi požadavky.
- Reprezentace: Data odesílaná nebo obdržená z REST API jsou často ve formě reprezentací zdrojů. To může zahrnovat JSON, XML, HTML nebo jiné formáty.



Obrázek 2.2: Příklad RESTful komunikace [19]

2.3 Popis aplikace

Aplikace umožňuje přihlášeným uživatelům vytvářet kreslící šablony (omalovánky) následujícím způsobem: uživatel nakreslí obrázek rozdělený do několika vrstev. Každá vrstva reprezentuje jednu barvu z výsledného obrázku a vrstvy jsou umístěny v pořadí určeném konkrétním autorem obrázku. Výsledný projekt vytváří kreslící šablonu, kde pořadí vrstev definuje kroky pro malování. Tuto šablonu autor nahrává na server, odkud ji další přihlášení uživatelé mohou stáhnout a postupně překreslovat obrázek vrstvu po vrstvě. Při tom je každá vrstva šablony zobrazena jako pozadí. Po dokončení všech kroků získá uživatel výsledný počet bodů na základě přesnosti kreslení všech kroků.

Aplikace také obsahuje knihovnu kreslících šablon, do které uživatelé mohou nahrávat vlastní šablony. Každá šablona má k dispozici informace, jako jsou název šablony, kategorie, informace o autoru, komentáře uživatelů, hodnocení, galerie výsledných obrázků od těch uživatelů, kteří tuto šablonu stáhli a překreslili.

2.4 Funkční požadavky

- **F1 - Přihlášení, Registrace:** Systém umožní uživatelům se přihlásit a zaregistrovat se.
- **F2 - Změna osobních informací:** Systém umožní uživatelům měnit své osobní a přihlašovací údaje.
- **F3 - Vytvoření šablony:**
  - **F3.1 - Kreslení vrstev:** Systém umožní uživatelům kreslit vrstvy obrázku a nastavit pro každou z nich barvu pro zobrazení samotné vrstvy pro daný krok a barvu, kterou jiní uživatelé budou používat pro kreslení v tomto kroku.
  - **F3.2 - Nahrávání obrázků:** Systém umožní uživatelům používat předem připravené png-soubory jako vrstvy šablony a nastavit barvy pro zobrazení vrstvy a kreslení.
  - **F3.3 - Uložení do knihovny:** Systém umožní uživatelům uložit novou šablonu do knihovny šablon. Název šablony musí být unikátní v seznamu všech šablon jednoho uživatele.
- **F4 - Odstranění šablony:** Systém umožní autorovi šablony odstranit ji z veřejné knihovny šablon, ale pouze v případě, že šablona má nulový počet stažení.
- **F5 - Přidání kategorie:** Systém umožní uživateli přidávat šablony do kategorií.
- **F6 - Stažení šablony:** Systém umožní uživateli přidávat šablony z veřejné knihovny do jeho vlastní knihovny pro pozdější překreslení.
- **F7 - Kreslení podle šablony:** Systém umožní uživatelům nakreslit obrázek krok za krokem podle vybrané šablony.
- **F8 - Uložení rozpracované kresby:** Během procesu překreslení systém umožní uživatelům uložit rozpracovanou kresbu. Následně, když uživatel bude chtít pokračovat v kreslení, systém načte dříve uložený stav.
- **F9 - Ukončení překreslení šablony:** Systém umožní uživatelům ukončit překreslování v libovolný moment během procesu kresby. Přitom systém přidělí body za provedené kreslení na základě přesnosti dodržení proběhlých kroků.

- **F10 - Sdílení výsledků:** Systém umožní uživatelům nahrávat výsledný obrázek do galerie kreseb u jednotlivých šablon.
- **F11 - Přidání komentáře:** Systém umožní uživateli přidávat veřejné komentáře na stránce šablony, uzivatele nebo obrazku v galerie.
- **F12 - Nahlášení:** Systém umožní uživateli nahlasit šablonu, komentář nebo jiného uživatele.
- **F13 - Sledování uživatele:** Systém umožní uživateli sledovat jiné uživatele.
- **F14 - Hodnocení šablony, komentáře, obrázku:** Systém umožní uživateli přidat „like“ nebo „dislike“ reakce jako hodnocení šablony, komentaře nebo obrázku v galerie u šablony.
- **F15 - Změna stavu nahlášení:** Systém umožní moderátorovi a administrátorovi uzavřít nahlášení nebo otevřít již uzavřené.
- **F16 - Správa kategorií:** Systém umožní administrátorovi vytvářet, mazat nebo editovat existující kategorie.
- **F17 - Správa uživatelských účtů:** Systém umožní administrátorovi vytvářet, editovat (včetně uživatelských rolí) nebo odstraňovat uživatelské účty ze systému.

Nice to Have:

- **F18 - Uložení jako koncept:** Během procesu vytváření šablony systém umožní uživateli uložit rozpracovanou šablonu, ale nezařadit ji do veřejné knihovny. Tato šablona bude dostupná pouze autorovi.
- **F19 - Verifikace šablony:** Systém umožní moderátorovi a administrátorovi ověřit šablonu. Po ověření systém přidá body za šablonu do celkového hodnocení autorovi a těm uživatelům, kteří již tuto šablonu překreslili.
- **F20 - Blokace šablony:** Systém umožní moderátorovi a administrátorovi zablokovat šablonu, pokud není přípustná podle pravidel platformy (například obsahuje nepovolený obsah). Šablona zůstane v systému, ale nebude přístupná pro uživatele.
- **F21 - Blokace uživatelského účtu:** Systém umožní moderátorovi a administrátorovi zablokovat uživatelský účet, pokud uživatel porušuje pravidla platformy. Účet stále bude existovat v systému, ale uživatel nebude schopen se přihlásit, a jeho profil nebude dostupný jiným uživatelům.
- **F22 - Správa šablon:** Systém umožní moderátorovi a administrátorovi editovat existující šablony jiných uživatelů.

- **F23 - Mazání obrázků v galerie:** Systém umožní moderátorovi a administrátorovi mazat obrázky, které uživateli přidali do galerii na stránce šablony.
- **F24 - Odblokování:** Systém umožní administrátorovi odblokovat šablony a uživatelské účty.

2.5 Nefunkční požadavky

- **N1 - Multiplatformnost:** Aplikace má být dostupná na různých platformách, včetně Androidu, iOS, Windows a ve webovém prohlížeči.
- **N2 - Vývoj Aplikace:** klientská aplikace musí být napsaná ve frameworku Flutter, který podporuje multiplatformní tvorbu.
- **N3 - REST API:** Aplikace musí komunikovat se serverem pomocí REST API.

2.6 Uživatelské scénáře

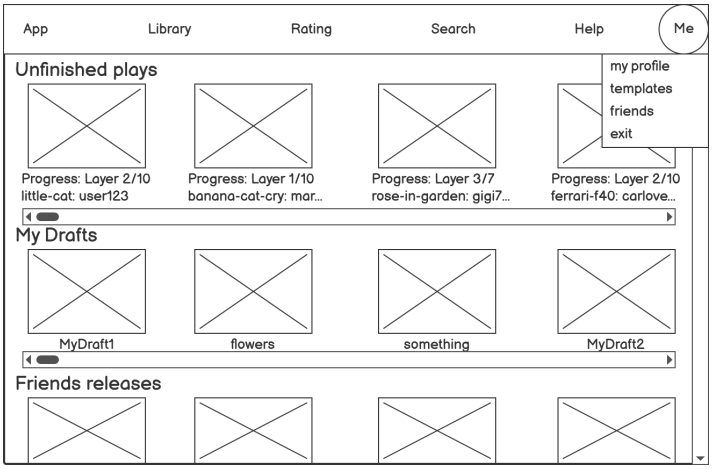
Případy použití podrobně popisují jednotlivé funkční požadavky s ohledem na instrukce nefunkčních požadavků. Případy použití jsou představeny pomocí scénářů, které modelují uživatelskou interakci.

UC1 - Vytvoření uživatelského účtu

Tento případ použití popisuje proces registrace, přihlášení a nastavení osobních údajů.

1. Uživatel zmáčkne tlačítko *„Registrace“* a následně systém zobrazí formulář pro vyplnění.
2. Uživatel zadá do odpovídajícího pole uživatelské jméno, e-mailovou adresu a heslo.
3. Systém zkontroluje, zda uživatelské jméno a e-mail již v systému neexistují.
   a. Pokud údaje v systému neexistují, systém vytvoří nový uživatelský účet s těmito údaji.
   b. Pokud údaje v systému existují, systém vrátí formulář pro změnu údajů.
4. Uživatel klikne na tlačítko *„Přihlásit se“*, zadá uživatelské jméno a heslo a následně proběhne přihlášení. Systém přesměruje uživatele na hlavní stránku.
5. Uživatel klikne na tlačítko *„Můj profil“* (Obrázek 2.3) a na této stránce stiskne *„Změnit údaje“*.

6. Uživatel zadá nové uživatelské jméno, systém zkontroluje, zda již neexistuje, a aktualizuje uživatelský účet.



Obrázek 2.3: Dostupné sekce pro uživatele s rolí „User“.

UC2 - Vytvoření šablony

Tento případ použití popisuje proces vytvoření nové šablony. Aplikace nabízí dva způsoby: kreslení a nahrání připravených PNG obrázků. Kromě toho existují dvě možnosti uložení šablony. Vzhledem k tomu jsou připraveny dva scénáře.

Scénář A.

1. Uživatel stiskne tlačítko *„Šablony“* ze seznamu dostupných akcí pro jeho roli. (Obrázek 2.4)

2. Uživatel stiskne tlačítko *„Nová šablona“*, a systém otevře okno, ve kterém uživatel zvolí možnost *„Nakreslit šablonu“*. (Obrázek 2.5) System spustí režim vytvoření šablony.

3. Uživatel zvolí barvu a nakreslí jednobarevný obrázek, který bude první vrstvou šablony. (Obrázek 2.6)

4. Uživatel stiskne tlačítko *„Přidat vrstvu“*, což změní oblast kreslení na *„Vrstva 2“*. Zvolí jinou barvu, a vše, co nyní kreslí, bude nad předchozí vrstvou.

5. Tímto způsobem nakreslí několik vrstev, kde každá je jednobarevná.

6. Uživatel stiskne tlačítko *„Uložit šablonu“*. Systém načte stránku se seznamem všech vrstev, kde u každé vrstvy je zobrazena její barva a velikost tužky. (Obrázek 2.7)

7. Pro každou vrstvu uživatel nastaví dvě barvy: jednu pro zobrazení vrstvy uživateli jako pozadí pro kreslení a druhou barvu pro kreslení.[1]

8. Uživatel přidá šablonu do několika kategorií ze seznamu existujících kategorií.

9. Uživatel stiskne tlačítko *„Uložit a přidat do knihovny šablon“*. Systém otevře okno, kde uživatel zadá jméno šablony.

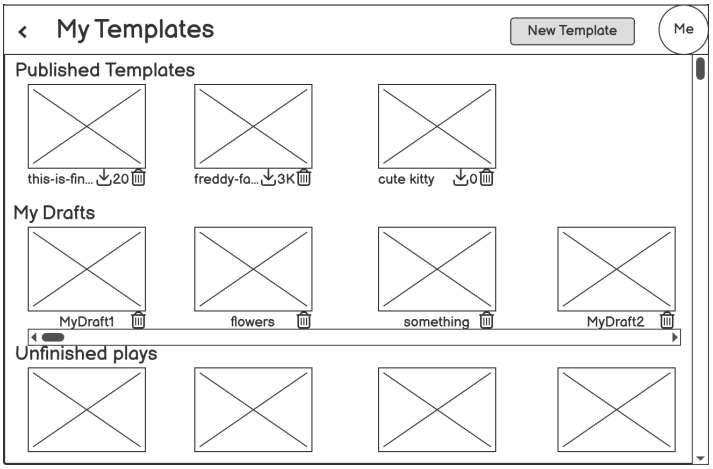
10. Uživatel zadá unikátní jméno pro seznam šablon vytvořených tímto uživatelem.

11. Stiskne tlačítko *„Uložit“*. Systém vygeneruje výsledný obrázek této šablony a uloží ho jako ukázkový. Tento obrázek uvidí uživateli ve knihovně šablon. Systém přidá šablonu do veřejné knihovny a nastaví maximální možný počet bodů za překreslení šablony. Po nahrání do veřejné knihovny není editace vrstev šablony již možná.

12. Uživatel přejde na hlavní stránku.

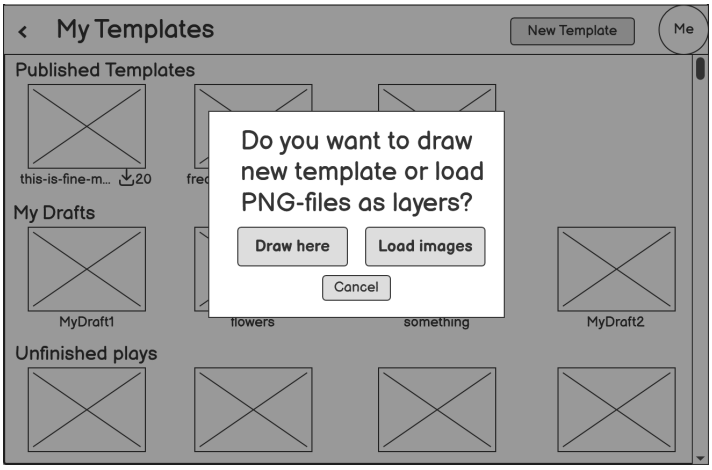
13. Stiskne tlačítko *„Šablony“* ze seznamu dostupných akcí pro jeho roli a systém načte všechny jeho šablony.

14. Vybere jednu ze svých šablon a stiskne *„Smazat“*. Pokud v okamžiku smazání má nulový počet stažení, systém šablonu odstraní.

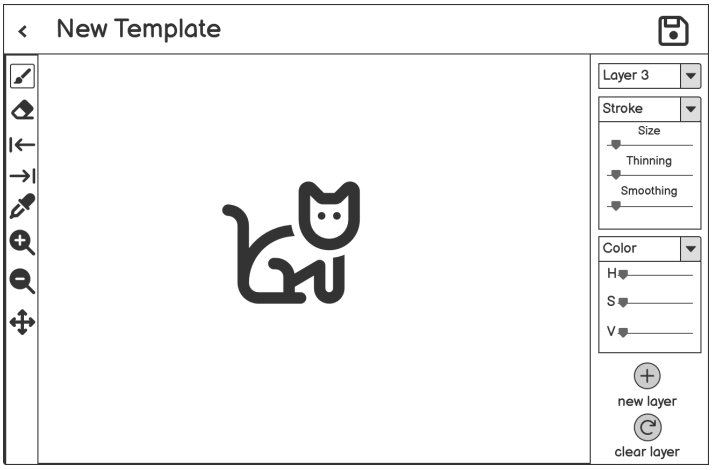


Obrázek 2.4: Seznamy šablon pro daného uživatele.Kromě zobrazených seznamu ještě jsou: seznam “Nově stažené šablony” a “Překreslené šablony”.

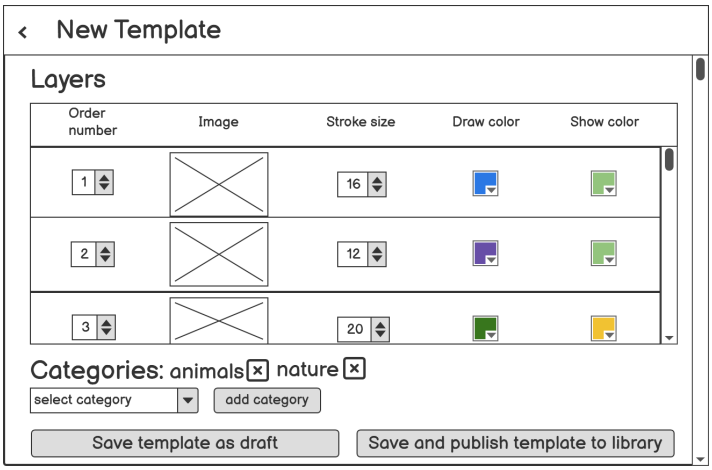
[1]Barvy musí být různé, aby uživatelé mohli snadno pochopit, co přesně mají překreslit. Například, uživatel má nakreslit kruh žlutou barvou. Originální vrstva slouží jako pozadí pro tuto kresbu a bude zobrazena červenou barvou, aby tvar kruhu byl přehledný. Při verifikaci šablony může moderátor změnit barvu zobrazení vrstvy na pozadí, pokud považuje originální barvu za nevhodnou.



Obrázek 2.5: Okno vyberu zpusobu vytvoreni sablony: Kreslit primo v aplikaci nebo nahrat PNG-soubory.



Obrázek 2.6: Kreslící režim.



Obrázek 2.7: Nastavení parametrů šablony a zpusoby uložení.

Scénář B.

1. Přihlášený uživatel stiskne tlačítko *„Nová šablona“*, a systém otevře okno, ve kterém uživatel zvolí *„Načíst soubory“*. (Obrázek 2.5) Systém otevře okno pro výběr souborů ze souborového systému zařízení.

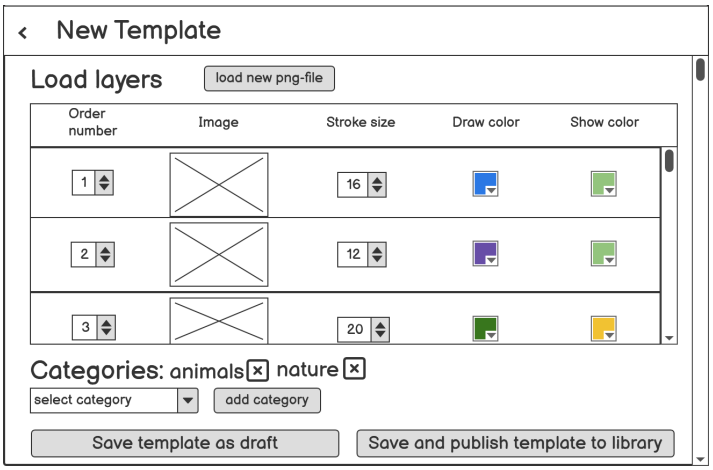
2. Uživatel nahraje připravené PNG obrázky a rozmístí je ve seznamu vrstev v požadovaném pořadí. (Obrázek 2.8)

3. U každé vrstvy uživatel nastaví dvě barvy: jedna barva ukazuje, jak bude zobrazena originalní vrstva jako pozadí při překreslení, druhá barva ukazuje barvu, kterou uživatel bude používat pro překreslení.

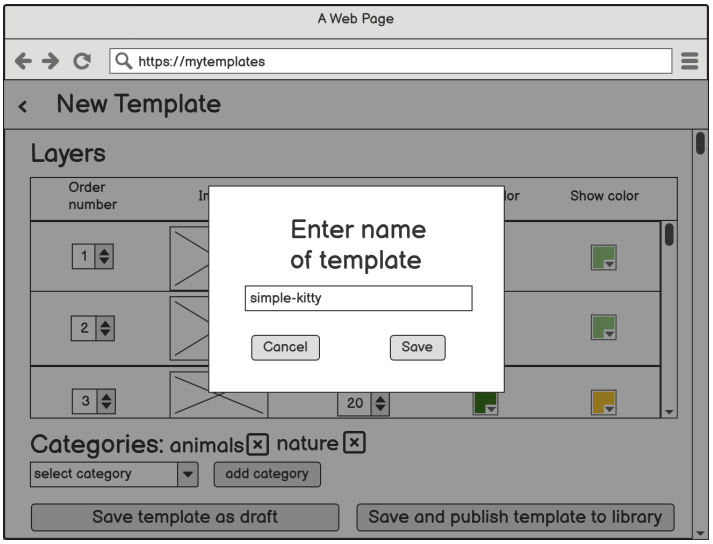
4. Uživatel stiskne tlačítko *„Uložit jako koncept“*. Systém otevře okno, ve kterém uživatel musí zadat název šablony. (Obrázek 2.9)

5. Uživatel zadá název šablony. Název musí být unikátní pro seznam šablon vytvořených tímto uživatelem.

6. Uživatel stiskne tlačítko *„Uložit“*. Systém uloží šablonu ve stavu *„Draft“*. Tato šablona bude dostupná pouze pro autora, který ji může editovat a následně uložit do veřejné knihovny.



Obrázek 2.8: Režim vytvoření šablony přes nahrání souboru.



Obrázek 2.9: Zadání unikátního názvu šablony.

UC3 - Překreslení šablony

Tento případ použití popisuje celý proces překreslení šablony.

1. Uživatel otevře veřejnou knihovnu šablon (Obrázek 2.10) a vybere jednu z nich. Systém načte detaily této šablony, zejména její název, ukázkový obrázek, galerii výsledných pokusů jiných hráčů o překreslení této šablony, komentáře, hodnocení a počet stažení. (Obrázek 2.11)
   a. Uživatel stiskne tlačítko *„Přidat do mé knihovny“*. V této chvíli se šablona přidá k seznamu šablon, které uživatel má rozpracované. Přitom uživatel může pokračovat ve výběru dalších šablon.
   b. Uživatel stiskne tlačítko *„Hrát“*, a systém otevře režim překreslování pro hráče. (Další scénář pokračuje s ohledem na tuto volbu).
2. Po otevření režimu překreslování uživateli se zobrazí první vrstva šablony jako pozadí, a uživatel ji překreslí. Momentálně je uživatel v prvním kroku hry. (Obrázek 2.12)
3. Uživatel stiskne tlačítko *„Další krok“*. Tím uživatel přejde do druhého kroku, systém odstraní předchozí pozadí a načte obrázek další vrstvy nad kresbou uživatele. Přitom editování předchozího kroku už není možné. (Obrázek 2.13)
4. Uživatel kreslí každý další krok stejným způsobem, počet kroků se rovná počtu vrstev u šablony.
5. Po překreslení všech vrstev uživatel stiskne *„Další krok“*. Jelikož žádná další vrstva už není, tento krok dovolí uživateli nakreslit cokoliv a jakoukoliv barvou na svém obrazku.

6. Uživatel stiskne tlačítko *„Ukončit“*. Systém vypíše počet bodů přesnosti, který uživatel dostal, a nabídne přidat výsledný obrázek do galerie obrázků u šablony. (Obrázek 2.14)

7. Uživatel stiskne *„Ano“*. Systém uloží obrázek.

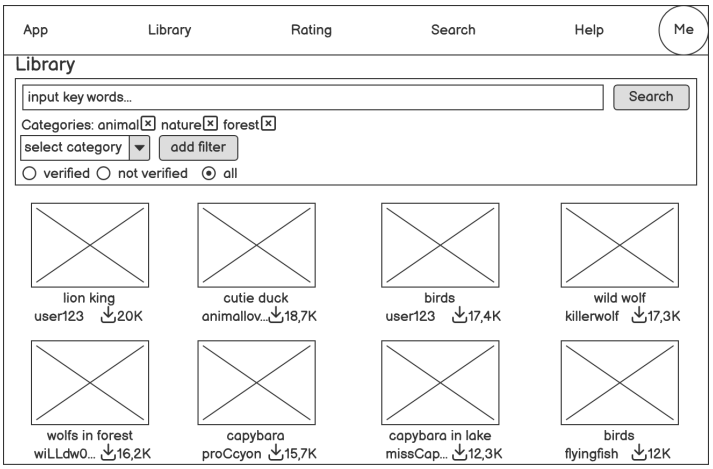
8. Uživatel otevře stránku šablony a uvidí v galerii svůj obrázek.

9. Uživatel přidá této šabloně značku *„Líbí se“* jako hodnocení a stiskne *„Přidat komentář“*.

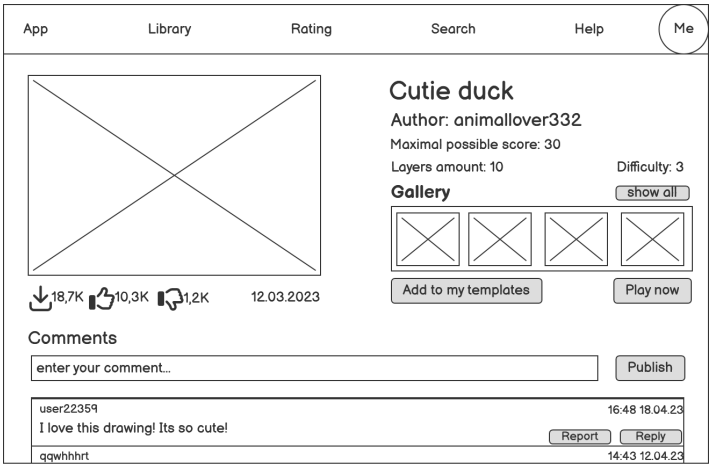
10. Uživatel vyplní pole komentáře a stiskne *„Přidat“*.

11. Na stránce šablony je odkaz na stránku autora. Uživatel otevře stránku autora a stiskne tlačítko *„Sledovat“*. (Obrázek 2.15) Tím se přidá autora do seznamu svých sledovaných uživatelů.

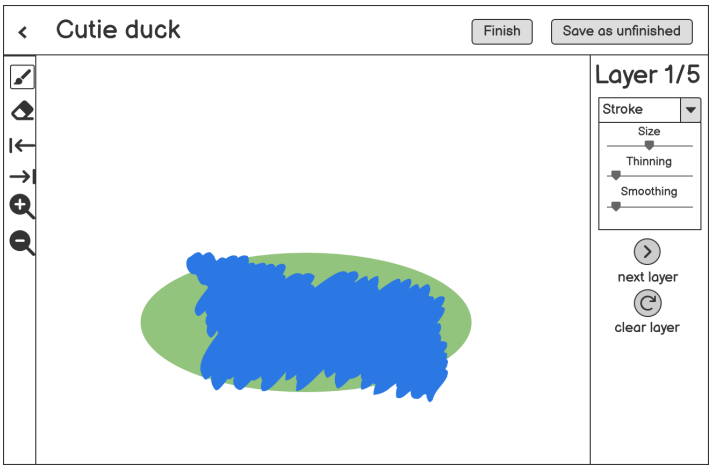
12. Uživatel otevře sekce *„Sleduju“* a zkontroluje, že nový uživatel se přidal do seznamu.



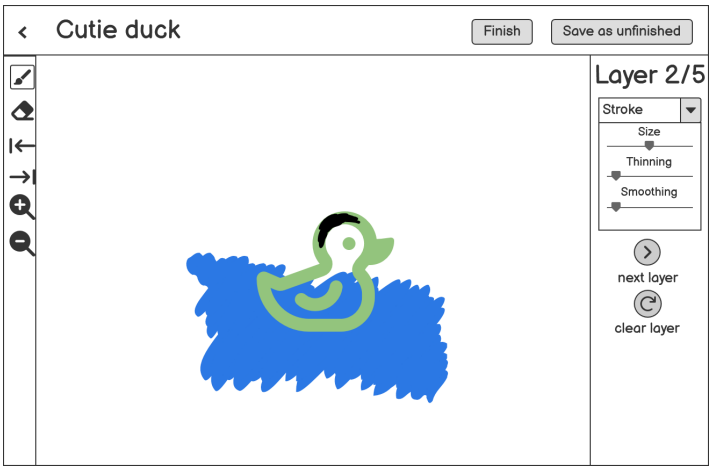
Obrázek 2.10: Veřejná knihovna šablon. Vyhledávání je možné podle kategorií.



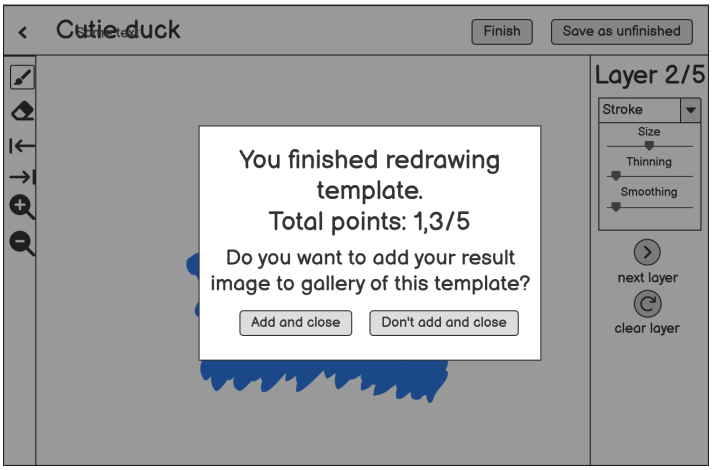
Obrázek 2.11: Informační stránka šablony.



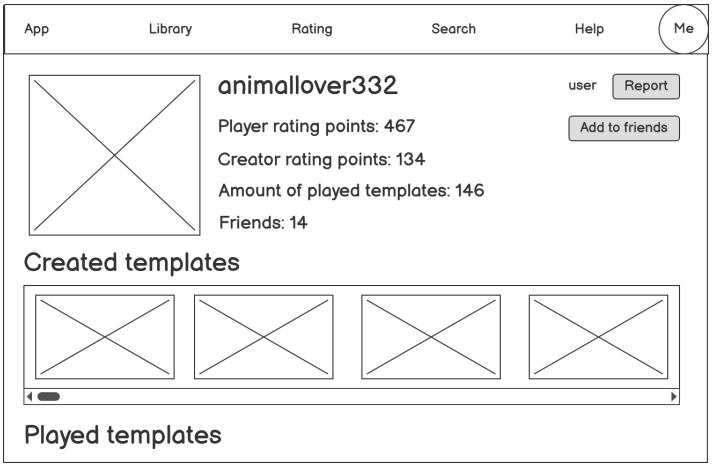
Obrázek 2.12: Ukázka prvního kroku v režimu překreslení.



Obrázek 2.13: Ukázka druhého kroku v režimu překreslení.



Obrázek 2.14: Ukončení překreslení.



Obrázek 2.15: Cizí uživatelský profil.

UC4 - Nahlášení

Tento případ použití popisuje proces nahlášení šablony.

1. Přihlášený uživatel otevře stránku nějaké šablony.
2. Uživatel stiskne tlačítko *„Nahlásit šablonu“*. Systém otevře okno vytvoření nového nahlášení a nabídne uživateli vybrat jedno z předdefinovaných odůvodnění nebo napsat vlastní do textového pole.
3. Uživatel napíše vlastní důvod nahlašení a stiskne *„Nahlásit“*. Systém vytvoří nové nahlášení.

UC5 - Blokace šablony

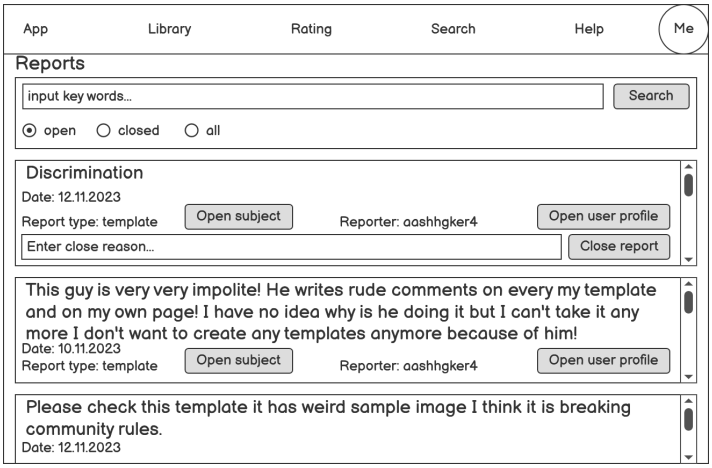
Tento scénář popisuje proces blokování šablony na základě nahlášení.

1. Moderator stiskne tlačítko *„Nahlášení“* a systém načte seznam všech otevřených nahlášení (Obrázek 2.16).

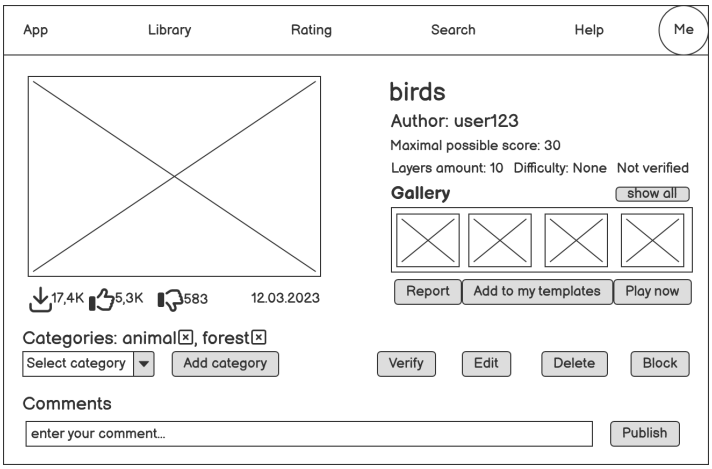
2. Moderator vybere jedno z nich a z tohoto okna přejde přímo na stránku šablony.

3. Na stránce šablony v ukázkovém obrázku uvidí obsah nepovolený dle pravidel platformy a stiskne tlačítko *„Zablokovat“* (Obrázek 2.17). Systém odstraní šablonu z veřejné knihovny, bude dostupná pouze pro moderátory a administrátory systému.

4. Moderator se vrátí do seznamu stížností, uzavře tu, kterou vyřešil, a napíše důvod uzavření.



Obrázek 2.16: Seznam nahlášení, které vidí moderator.



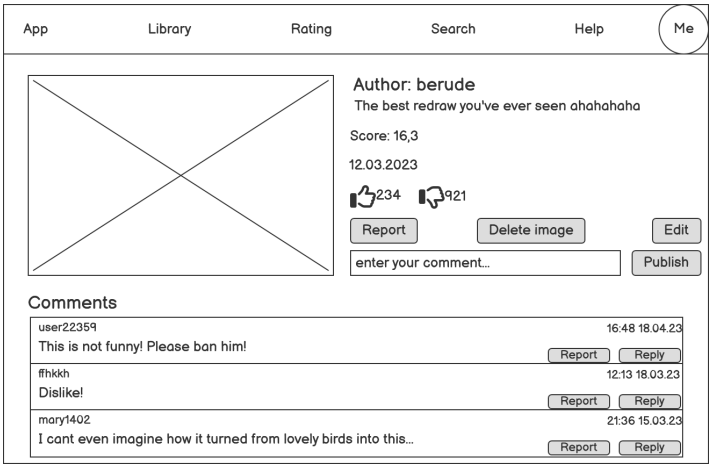
Obrázek 2.17: Přehled stránky šablony z účtu moderatora.

UC6 - Verifikace šablon

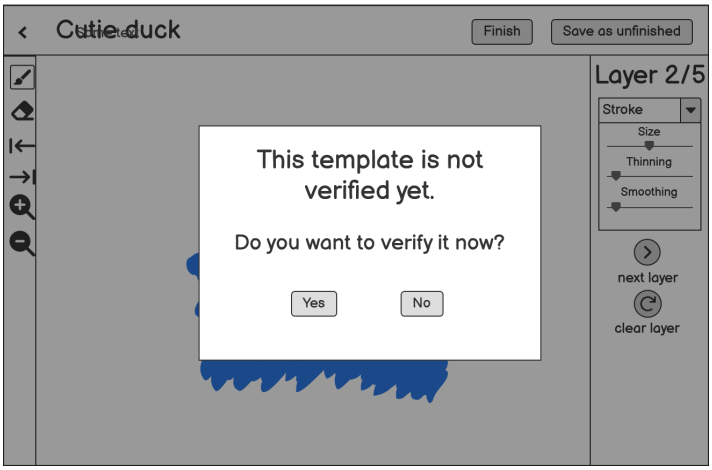
Tento scénář popisuje proces verifikace šablony moderátorem. Za překreslení verifikované šablony uživatel získává body, které se mu připočtou do globálního hodnocení.

1. Moderátor stiskne tlačítko *„Verifikace šablon“*. Systém načte stránku, na které jsou neověřené šablony, seřazené podle data vytvoření.
2. Moderátor vybere jednu šablonu a načte se jeji stránka.
3. Moderátor stiskne *„Zobrazit galerii obrázků“* a systém načte všechny uživatelské obrázky pro tuto šablonu.
4. Moderátor uvidí obrázek s nepovoleným obsahem (Obrázek 2.18), přejde na stránku autora obrázku a stiskne *„Zablokovat uživatele“*. Systém zablokuje uživatelský účet, a uživatel nebude schopen přihlásit se do systému, uvidí oznámení, že jeho účet byl zablokován.
5. Moderátor se vrátí k obrázku a stiskne *„Smazat obrázek“*. Systém odstraní obrázek.
6. Moderátor má dva způsoby verifikace šablony:
   a. Moderátor stáhne šablonu pro překreslení jako obyčejný uživatel. Dále scénář pokračuje týmto výběrem.
   b. Na stránce šablony moderátor může stisknout tlačítko *„Verifikovat“*. Tim se přeskočí etapu překreslení šablony.
7. Následně moderátor stáhne šablonu pro překreslení jako obyčejný uživatel. Poté následuje celý proces překreslení šablony jako v 3. případu užití, až po bod 7. V tomto kroku systém nabídne moderátorovi otevřít režim verifikace šablony. (Obrázek 2.19)
8. Moderátor stiskne tlačítko *„Verifikovat“* a přejde na stránku verifikace (Obrázek 2.20). Systém načte stránku se seznamem všech vrstev, kde u každé vrstvy jsou zobrazeny dvě barvy: jedna pro zobrazení vrstvy uživateli jako pozadí pro kreslení a druhá barva pro kreslení. Moderátor může změnit barvu pozadí vrstvy, pokud považuje původní barvu za nevhodnou.
9. Moderátor změní barvu u několika vrstev tak, aby šablona byla lépe viditelná pro uživatele.
10. Následně moderátor vybere složitost šablony. Existují tři složitosti: jednoduchá (1), střední (2) a těžká (3). Moderátor vybere střední složitost.
11. Moderátor změní kategorii u šablony, protože za dobu čekání na verifikaci v systému se přidali nové kategorie, které vyhovují tématu šablony.

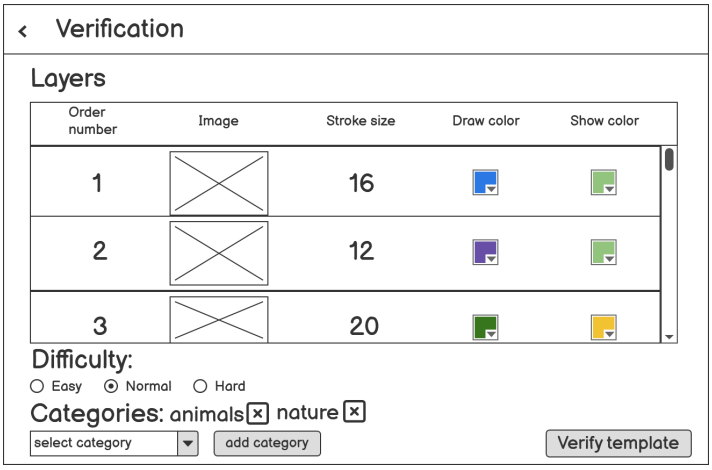
12. Moderátor stiskne tlačítko *„Dokončit verifikaci“* a systém změní stav šablony na verifikovaný. Všichni uživatelé, kteří si šablonu již překreslili, obdrží body za překreslení, vynásobené složitostí šablony. Body se přičtou k jejich globálnímu hodnocení a globální hodnocení autora šablony se také zvýší o stejný počet bodů.



Obrázek 2.18: Informační stránka uživatelského obrázku v galerii.



Obrázek 2.19: Systém nabízí moderátoru verifikovat šablonu.



Obrázek 2.20: Režim verifikace šablony.

UC7 - Odblokování

Tento případ popisuje proces odblokování uživatelského účtu administrátorem a změnu uživatelského jména z důvodu porušení pravidel společenství.

1. Administrator otevře seznam zablokovaných uživatelů.
2. Administrator zadá uživatelské jméno nebo email uživatele, kterého potřebuje odblokovat. Vedle jména zablokovaného uživatele stiskne tlačítko *„Odblokovat“*.
3. Následně přejde na stránku odblokovaného uživatele a stiskne tlačítko *„Editovat účet“*. Systém načte okno editace uživatelských údajů.
4. Administrator změní uživatelské jméno tohoto uživatele a stiskne *„Uložit“*. Systém uloží změny a zašle email uživateli s novými přihlašovacími údaji.

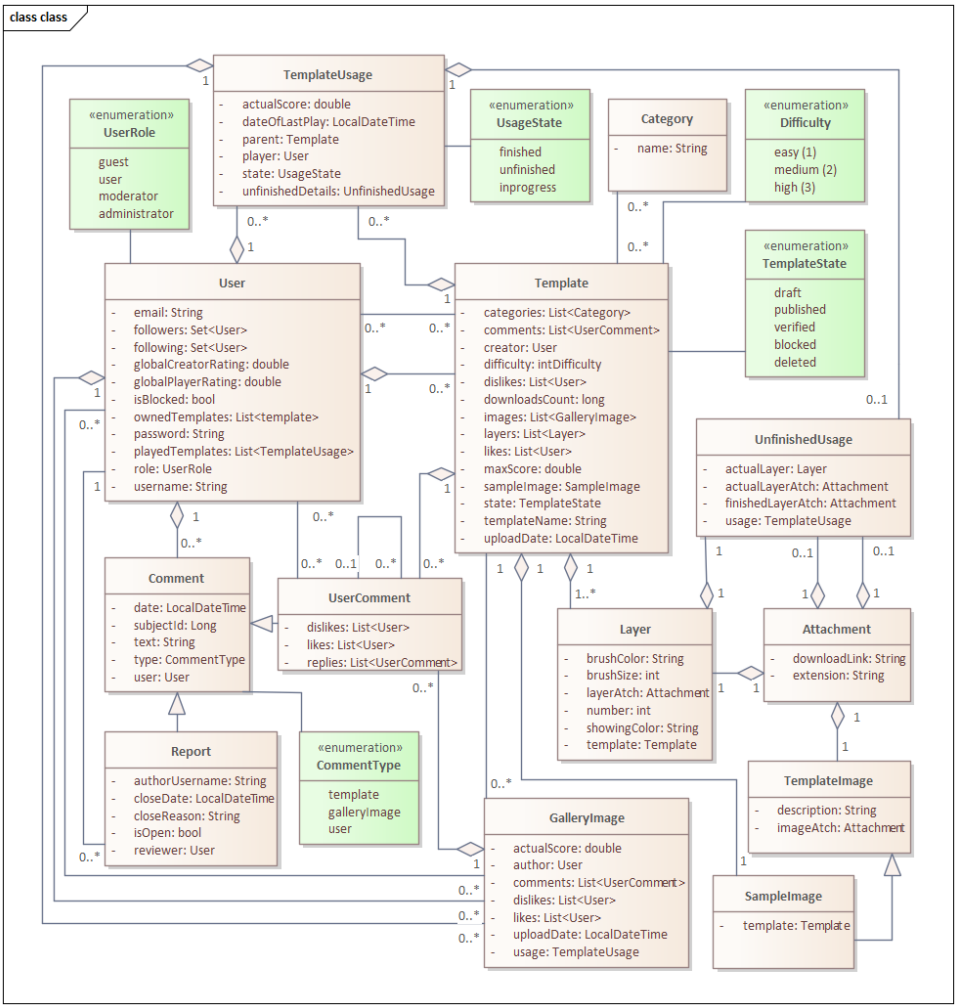
UC8 - Správa kategorií

1. Administrator stiskne tlačítko *„Kategorie“*. Systém načte seznam existujících kategorií.
2. Administrator klikne na tlačítko *„Vytvořit novou kategorii“* a systém načte okno pro vytvoření.
3. Administrator zadá název kategorie a stiskne *„Uložit“*. Pokud je název kategorie unikátní, systém uloží novou kategorii.

Kapitola 3

Návrh

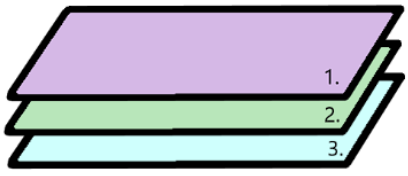
3.1 Datový model



Obrázek 3.1: Datový model aplikace.

Některé netriviální třídy a atributy z obrázku 3.1:

- **Template (Šablona):**
  - Atribut *maxScore* reprezentuje maximální možný počet bodů, které uživatel může získat za překreslení této šablony.
- **TemplateUsage (Využití šablony)** - objekt, který vznikne, když uživatel stáhne nějakou šablonu, a je specifický pro daného uživatele:
  - Atribut *actualScore* - aktuální počet bodů, které konkrétní uživatel získal v konkrétním datu (*dateOfLastPlay*) za překreslení této šablony. Pokud uživatel znovu překreslí stejnou šablonu a počet bodů bude větší, aktualizují se *actualScore* a *dateOfLastPlay*.
  - Atribut *player* - konkrétní uživatel, který stáhl šablonu.
  - Atribut *parent* popisuje šablonu, ke které se vztahuje objekt **TemplateUsage**.
  - Atribut *state* - překreslení šablony může být ukončeno (*finished*), když uživatel překreslí všechny vrstvy, nebo neukončeno(*unfinished*), pokud uživatel uloží rozpracovanou šablonu pro ukončení později (atribut **UnfinishedUsage** *unfinishedDetails*). Když proces překreslení právě probíhá, nebo když byla šablona přidána do uživatelské knihovny pro pozdější překreslení, stav objektu se změní na *in progress*.
- **UnfinishedUsage (Rozpracováná šablona, Obrázek 3.2):**
  - Atribut *actualLayer* - Obsahuje informace o tom, na které vrstvě uživatel uložil rozpracované překreslení.
  - Atribut *finishedLayerAtch* - odkaz na soubor, který reprezentuje uživatelské kreslení již vypracovaných vrstev této šablony.
  - Atribut *actualLayerAtch* - odkaz na soubor, který reprezentuje rozpracovanou vrstvu, na které právě uživatel zastavil kreslení.



1. actualLayerAtch
2. actualLayer.layerAtch
3. finishedLayerAtch

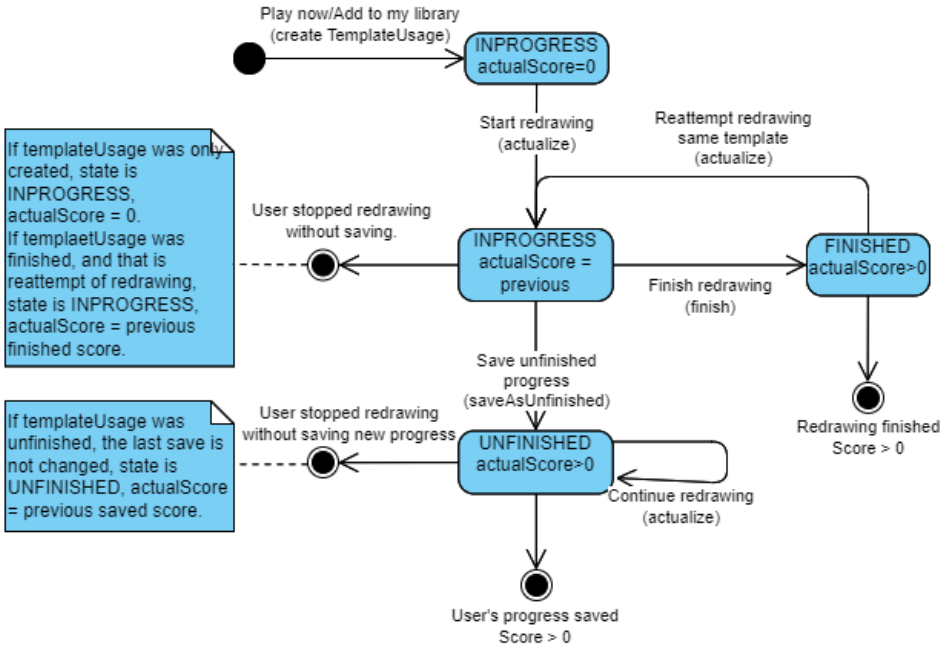
Obrázek 3.2: Schematická ukázka UnfinishedUsage.

- **Layer (Vrstva)** - objekt, který reprezentuje jednu vrstvu šablony a definuje nastavení barev a velikosti štětce:

- Atribut *number* - pořadové číslo vrstvy v šabloně.
- Atribut *layerAtch* - odkaz na soubor, který obsahuje zobrazení dané vrstvy.

- **TemplateImage (Obrázek, který patří k šabloně)** - abstraktní třída, obsahující odkaz na soubor. Má dva potomky: **SampleImage** a **GalleryImage**.
- **SampleImage (Ukázkový obrázek)** - Reprezentuje ukázkový obrázek šablony. Ukazuje, jak bude vypadat obrázek po překreslení šablony.
- **GalleryImage (Obrázek z galerie)** - Reprezentuje výsledný obrázek kresby konkrétního uživatele, který je uložen do galerie výsledků u jednotlivé šablony. Tento objekt je spojen s objektem **TemplateUsage**, protože vzniká, když uživatel dokončí kresbu.
- **Comment (Komentář)** - reprezentuje komentáře jednotlivých uživatelů. Má dva potomky:
  - **UserComment (Uživatelský komentář)** - Reprezentuje veřejné uživatelské komentáře u šablon, obrázků, atd.
  - **Report (Nahlášení)** - Reprezentuje uživatelské nahlášení. Seznam nahlášení je dostupný pouze pro moderátory a administrátory systému.

3.1.1 Diagram změny stavů objektu typu TemplateUsage



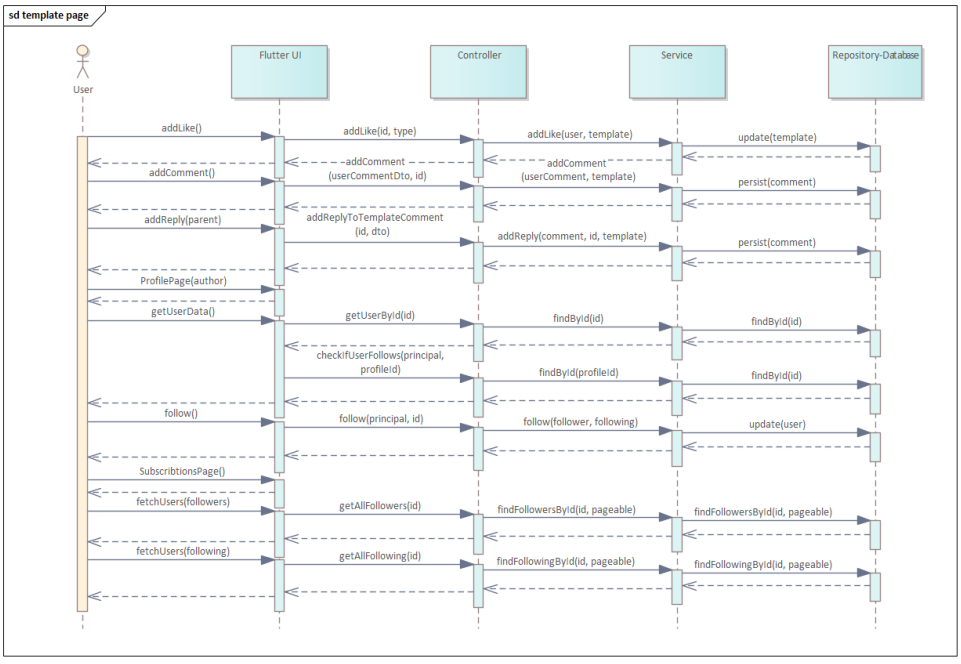
Obrázek 3.3: Stavy TemplateUsage.

Diagram 3.3 popisuje stavy, kterými může projít objekt TemplateUsage během procesu překreslení šablony.

3.2 Sekvenční diagramy

V této podkapitole jsou představeny sekvenční diagramy, které popisují některé uživatelské scénáře z pohledu vlivu uživatelských vstupů na reakce systému. V jednotlivých diagramech je zobrazena interakce uživatele s uživatelským rozhraním, komunikace mezi frontendem a vrstvou Spring Boot controllerů ze strany serveru. Dále je zobrazeno použití „byznys“ logiky. Pro zjednodušení v diagramu jsou vrstvy JPARepository a databáze spojené do jedné „lifeline“, protože většina příkazů používaných k volání metod vrstvy Repository a databáze mají stejný název.

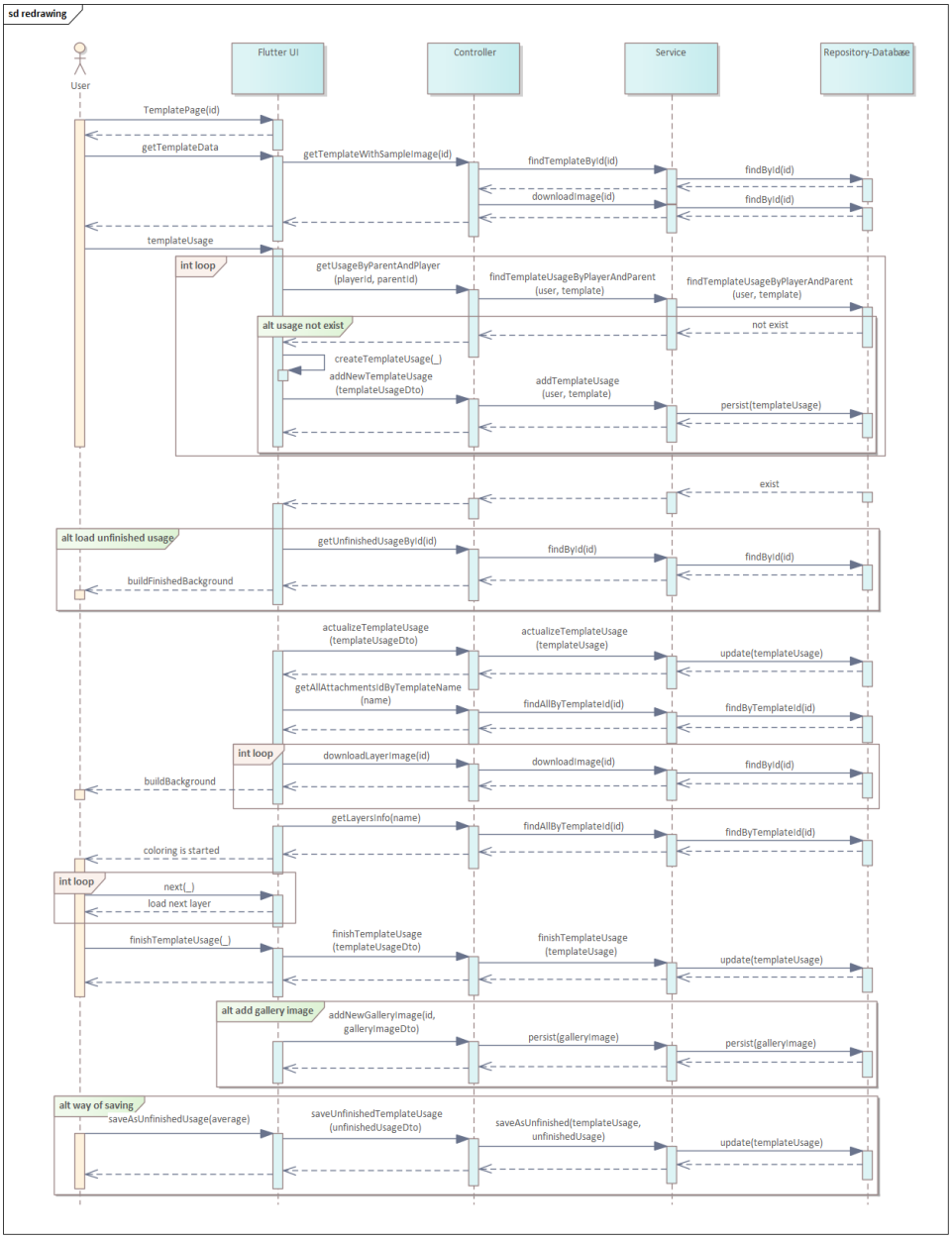
3.2.1 Interakce uživatele na stránkách knihovny šablon



Obrázek 3.4: Stavy TemplateUsage.

Diagram 3.4 představuje část průchodu uživatelským scénářem UC3, která zahrnuje interakce, které může uživatel provádět na stránce jednotlivých šablon (například *addLike*, *addComment*) a stránce jiných uživatelů (*follow*). Dále je zobrazeno načítání stránky se seznamy sledujících a sledovaných profilů jednotlivým uživatelem.

3.2.2 Proces překreslení šablony uživatelem



Obrázek 3.5: Stavy TemplateUsage.

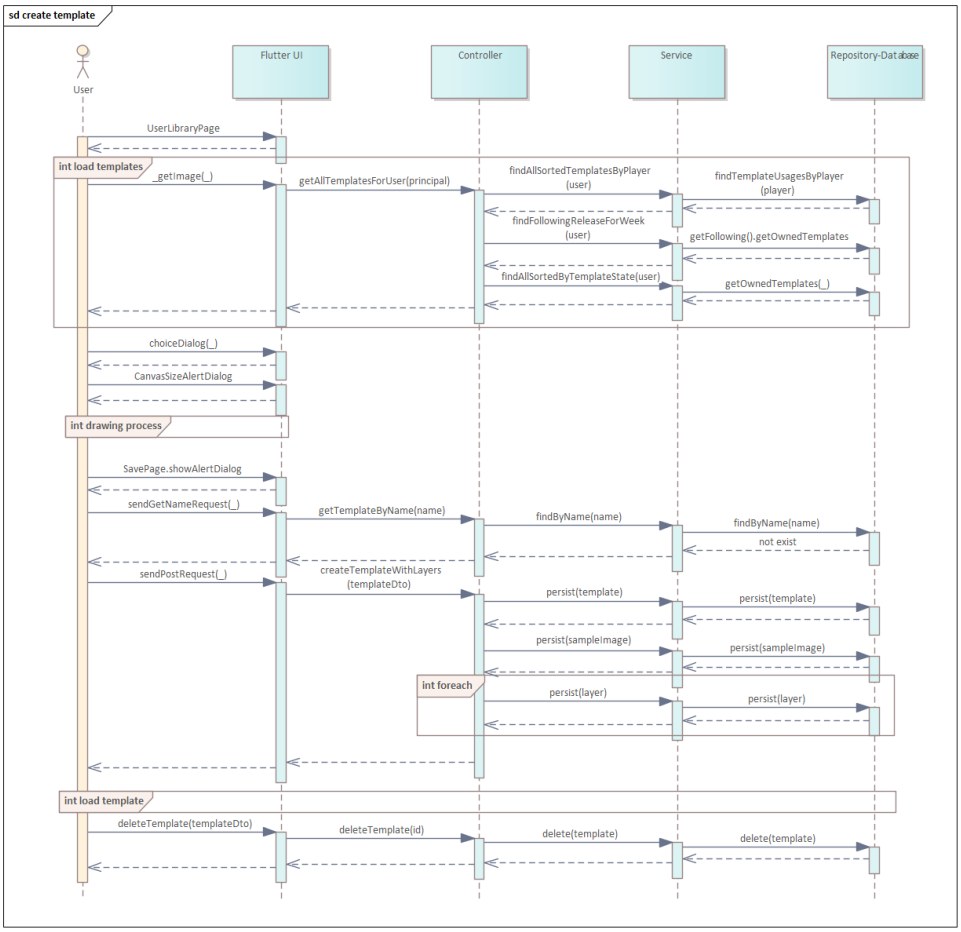
Diagram 3.5 popisuje proces překreslení od výběru šablony až po ukončení samotného překreslení. Diagram se vztahuje k první části případu užití UC3, která se zabývá právě překreslením.

Alternativní sekce popisují buď zcela odlišný způsob interakce nebo doplňkovou akci, která neovlivňuje další příkazy. Jako příklad prvního významu alternativy může uživatel buď ukončit překreslení (*finishTemplateUsage*) nebo uložit šablonu jako nedokončenou (*saveAsUnfinished*), nikoliv udělat obě

akce najednou. Jako doplňková akce v diagramu je představena sekce přidání obrázku do galerie (*alt add gallery image*).

Sekce *„int loop“* označují systémové provedení iterace, která může záviset na vnějších podmínkách. Například loop *next()* znamená načítání další vrstvy pro překreslení, kde akce přesunu do dalšího kroku provádí uživatel.

3.2.3 Vytvoření šablony

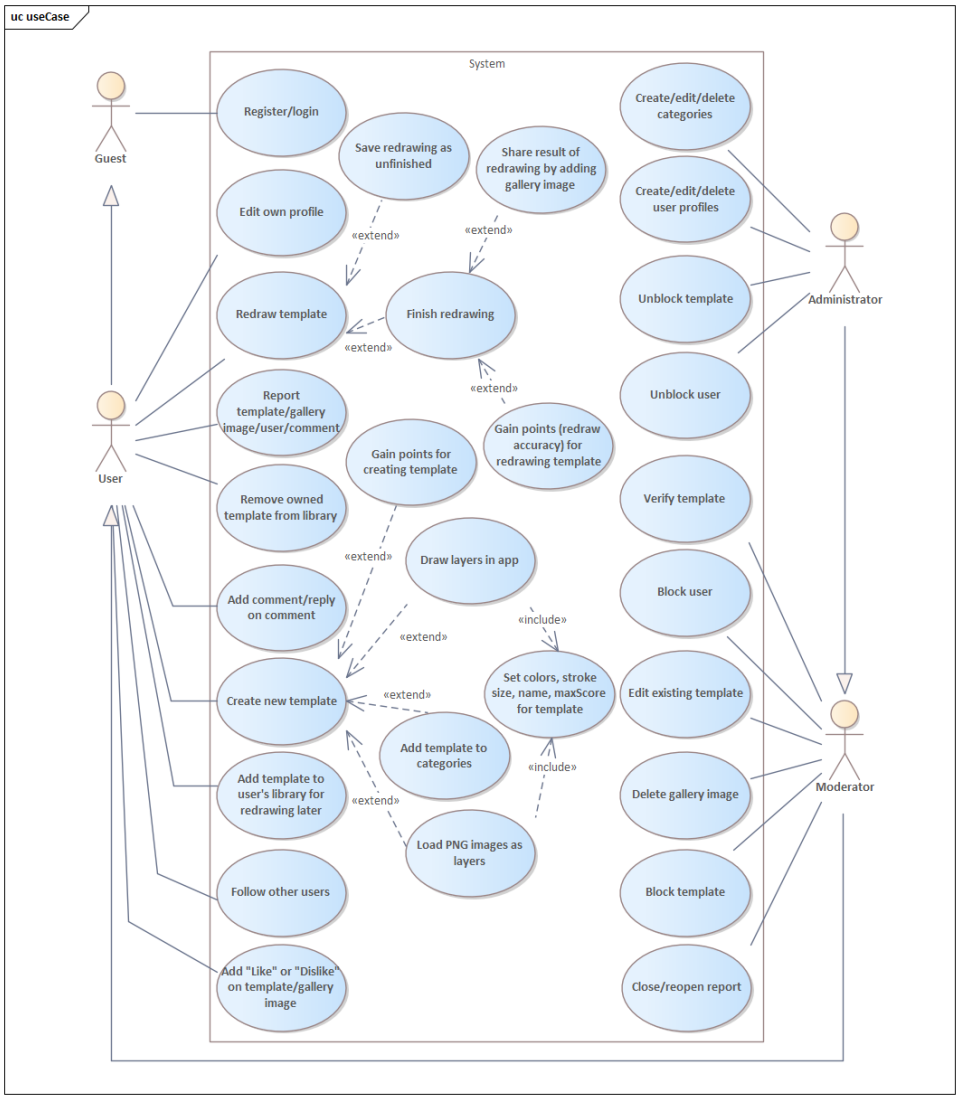


Obrázek 3.6: Stavy TemplateUsage.

Diagram 3.6 popisuje uživatelský scénář UC2. V daném procesu se dvakrát opakuje sekce *„load templates“*, přestože scénář zahrnuje krok odstranění šablony. V druhém případě nebyl plně zkopírován proces načítání dat všech potřebných šablon, pouze byla přidána ukázková sekce, která má stejný význam jako její plná verze.

Kromě toho byla pro zjednodušení ukázky zkrácena část interakce uživatele s uživatelským rozhraním během kreslení. Jsou zde představeny pouze důležité interakce a přesměrování. Celková kreslící sekce je označena jako *„int proces kreslení“*.

3.3 Role v systému



Obrázek 3.7: Use Case diagram.

Jak je vidět z diagramu 3.7, systém je rozdělen do tří hlavních rolí. Role Guest se nepovažuje za plnohodnotnou roli, protože představuje anonymního uživatele. Ten může prohlížet knihovnu, uživatelské účty, obrázky, komentáře, ale jinak nemůže interagovat s aplikací, dokud není přihlášen.

User

Role zahrnuje většinu dostupných funkcí. Je to základní role, kterou dostane uživatel po přihlášení do systému.

Moderator

Tato role zahrnuje všechny funkce role User a zároveň obsahuje část administrativních funkcí. Hlavním zaměřením této role je verifikace šablon a správa nahlášení.

Administrator

Tato role zahrnuje všechny dostupné funkce aplikace. Uživatelé s touto rolí jsou správci systému.

Kapitola 4
Implementace

4.1 Serverová část

Pro implementaci serverové části aplikace byl zvolen framework Spring Boot. V dané práci byla použita verze frameworku 2.7.3.

4.1.1 Architektura

Pro danou aplikaci byla zvolena monolitická architektura. Důvodem výběru je to, že pro potřeby dané aplikace je toto řešení vhodné: aplikace není velká a objekty mezi sebou mají velké množství interních volání, a proto výkon je lepší při použití monolitické architektury. Kromě toho je takový výběr vhodnější z hlediska časové náročnosti vývoje serverové části v rámci dané práce.

Taková aplikace je vyvinuta jako jeden balíček, používá jednu databázi pro všechny objekty. Tato architektura je vhodná pro vývoj menších aplikací, a protože je zabalena do jednoho balíčku, nasazení není tak složitý proces jako u aplikací se složitější architekturou.

4.1.2 Struktura

Aplikace má následující strukturu vrstev (balíčků):

Controllers

Tento balíček představuje vrstvu, která zpracovává HTTP požadavky přicházející z klientské strany. Požadavky mohou obsahovat klientská data ve formátu JSON nebo MultipartFile. Vrstva mapuje příchozí data do vnitřních entit, které následně jsou předány do byznys vrstvy na základě uživatelského požadavku. Kromě toho, v této vrstvě výsledky byznys vrstvy jsou zabaleny do objektu ResponseEntity, který se následně odesílá klientovi.

Service

Byznys vrstva zpracovává data dle logiky aplikace a přidává zpracovaná data do persistenční vrstvy. Obsahuje validace vstupních dat. Zároveň slouží jako

validační nadstavba pro autorizace.

Repository

Představuje vrstvu aplikace, která slouží ke komunikaci s databází. Jako implementace této vrstvy byla vytvořena rozhraní, která dědí od rozhraní JPARepository. Toto rozhraní umožňuje zjednodušit vývoj vrstvy přístupu k datům, protože poskytuje sadu standardních metod pro práci s databázemi a také umožňuje přidávat vlastní dotazy bez přímého psaní SQL kódu.

Model

Balíček obsahuje všechny entity aplikace, popisuje datové typy, anotace, názvy databázových tabulek a vztahy mezi entitami, realizované přes JPA anotace.

Config

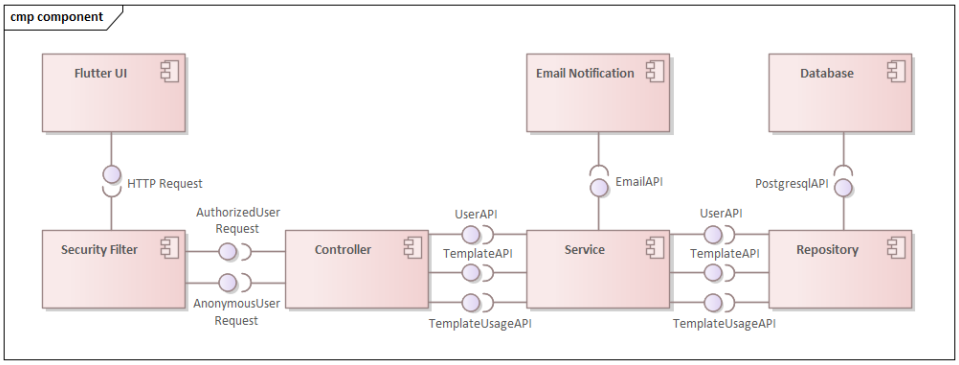
V daném balíčku jsou uloženy konfigurační soubory pro Spring Boot aplikaci. Obsahuje konfigurační soubor pro celkové nastavení aplikace, bezpečnostní a databázové konfigurace.

Security

Balíček obsahuje třídy, které se používají pro autentizaci do systému. Obsahuje autentizační token, zpracování uživatelských přihlašovacích údajů, porovnání hašovaných hesel, filtraci JWT.

Ostatní

Ve zbytku aplikace jsou doplňkové balíčky aplikace: Exceptions, Utils, Dto. Exceptions obsahuje vlastní chybové hlášky aplikace, Utils má doplňující funkcionalitu, Dto (data transfer object) představuje balíček tříd, které se používají k přenosu dat mezi klientem a serverem.



Obrázek 4.1: Diagram komponent.

Celkovou strukturu aplikace ve zjednodusene podobe lze vidět z diagramu 4.1.

4.1.3 JWT

JWT (JSON Web Token) se v aplikaci používá k autentizaci uživatele, což systému umožňuje ověřit identitu uživatele, který token předložil. Kromě toho se JWT využívá i k autorizaci, protože na základě obsahu tokenů systém určuje, ke kterým zdrojům má uživatel přístup. Toto probíhá prostřednictvím filtrování všech příchozích požadavků a určení, zda je uživatel autorizován. Pokud uživatel je držitelem nějakého tokenu, ověřuje se jeho platnost a příslušná uživatelská práva.

4.2 Klientská část

Klientská aplikace byla vytvořena ve frameworku Flutter. Flutter projekt obsahuje konfigurační soubor pubspec.yaml, v němž jsou definovány verze, SDK, knihovny a externí moduly.

4.2.1 Externí knihovny

Pro danou aplikaci byly použity následující netriviální knihovny:

```
dependencies:
 flutter:
   sdk: flutter
 flutter_hooks: ^0.20.1
 perfect_freehand:
   git:
     url: https://github.com/astrynx/perfect-freehand-dart.git
     ref: main
 flutter_colorpicker: ^1.0.3
 dio: ^5.3.3
 cyclop: ^0.7.0
```

Listing 4.1: Knihovny

- **Flutter Hooks** [20] - knihovna, která poskytuje nový druh widgetu - HookWidget, inspirovaný implementací React Hooks. Takové widgety umožňují využívat hooky pro správu stavu, efektů a dalších aspektů bez nutnosti vytváření třídních komponent. Toto činí kód čistším a snadněji udržitelným. Hooky lze snadno sdílet mezi různými komponentami, což usnadňuje opětovné použití logiky bez potřeby zásadního zásahu do architektury komponent. V projektu jsou takové widgety použity pro vyzkoušení jejich funkčnosti a protože na pevných místech aplikace byl ve srovnání se StatefulWidget HookWidget užitečnější.

- **Perfect freehand [21]** - knihovna, která podporuje rozpoznání tlaku pera na grafickém tabletu. Princip funkce této knihovny spočívá v tom, že bod na plátně obsahuje informace nejen o své pozici, ale také informace od senzoru o tlaku tužky v okamžiku, kdy byl nakreslen. Knihovna vypočítá obrys původní křivky vzhledem k informacím o pozici a tlaku. Kromě toho je částečně simulováno chování tlaku pro kreslení pomocí myši.

  V projektu nebyla použita aktuální verze této knihovny, protože vydaná knihovna obsahuje chybu, která způsobuje vytvoření špatného obrysu křivky při kreslení z tlakově citlivých zařízení. Kvůli této chybě vypadá kreslení s tužkou citlivou na tlak stejně jako kreslení s myší. Autor knihovny opravil tuto chybu ve zdrojovém kódu, ale nevydal novou verzi knihovny, proto připojení k projektu je možné pouze přes github repozitář. Nicméně, zdrojový kód knihovny byl zkopírován (fetch) do soukromého github repozitáře pro zabránění rozbití projektu v případě, že autor knihovny aktualizuje původní repozitář.

- **Flutter_colorpicker [22]** - knihovna, která umožňuje provádět zajímavý a podrobný výběr barvy pro kreslení. V projektu je použitý předpřipravený náhled výběru barvy poskytnutý autorem knihovny.

- **Dio [23]** - knihovna nezbytná k vylepšení komunikace pomocí protokolu HTTP se serverem, zejména pro vytváření objektů FormData a MultipartFile. FormData umožňuje vytvořit data pro HTTP POST požadavek, který obsahuje formulářová data. Knihovna poté zajistí, že tato data budou správně zakódována a odeslána jako součást HTTP požadavku. MultipartFile se používá pro přípravu a odesílání souborů v rámci HTTP požadavku jako součást objektu FormData. V projektu jsou tyto objekty používány k odesílání vrstev s obrázky na server.

- **Cyclop [24]** - knihovna, která poskytuje Eyedropper Tool. V projektu se používá v malovacích třídách pro získání barevní hodnoty tečky na plátně.

4.2.2 Struktura aplikace

Struktura projektu je rozdělena do 5 částí:

- Balíček **drawing** obsahuje třídy, které se týkají přímo procesu kreslení nebo vytváření šablon v aplikaci. Daná složka zahrnuje proces vytvoření šablony (třídy *DrawModePage* a *DrawingPage*), překreslení šablony (*RedrawModePage*, *ColoringPage*). Třída *BasePage* je abstraktní a slouží jako předek pro třídy *DrawingPage* a *ColoringPage*. Třída *CanvasSizeAlertDialog* se používá pro výběr formátu plátna při vytvoření nové šablony. *LoadPngPage* představuje proces vytvoření šablony prostřednictvím nahrání předem připravených PNG obrázků. Nakonec, třída *SavePage* slouží k uložení šablony na server.

- V balíčku **painters** jsou umístěny kreslicí třídy, které se používají buď pro konverzi kreslení do PNG-obrázku, nebo pro samotné kreslení na plátně. Tyto třídy dědí od třídy *CustomPainter*, která ve frameworku Flutter slouží k vytvoření vlastního grafického obsahu.

- Balíček **templatesLibrary** obsahuje zbytek viditelných widgetů aplikace a logiku klientské strany. Například třída *TemplatePage* představuje stránku obsahující informace o šabloně: ukázkový obrázek, informace o autorovi, popis, komentáře, obrázky, které jiní uživatelé sdíleli po překreslení dané šablony. Také stránka dovoluje uživateli začít proces překreslování nebo přidat šablonu do vlastní knihovny pro pozdější překreslení.

- **model** obsahuje datové modely, převážně používané pro uložení dat získaných z odpovědí serveru.

- V balíčku **utils** jsou pomocné třídy. Třída *AppBar* poskytuje hlavní hlavičku aplikace. *SessionManager* slouží pro uložení JWT, který se přeposílá serveru v hlavičce požadavku jako způsob prokázání autority. Dále balíček obsahuje pomocné třídy pro uložení a nastavení kreslicí křivky a tužky (třídy *Stroke*, *StrokeOptions*, *StrokePair*).

4.2.3 Vytvoření šablony

Většina procesu vytvoření šablony v aplikaci se odehrává v třídách *DrawModePage* a *DrawingPage* (Obrázek 4.2). *DrawModePage* představuje widget-předek, který uchovává nastavení pro uživatelské kreslení, jako je například výběr tužky a gumy, přiblížení, kroky zpět a vpřed, odstranění obsahu šablony, nastavení tužky/gumy, výběr barvy a přepínání mezi vrstvami kresby.

DrawingPage je widgetem následníkem, který představuje kreslící plátno. Z pohledu uživatele tento widget obsahuje pouze samotné plátno, na kterém probíhá kreslení, ale z pohledu systému tento widget zahrnuje většinu logiky kreslení. Oba widgety mezi sebou komunikují pro předání uživatelských nastavení. Nicméně, nakreslené křivky, použité barvy a seznam vrstev uchovává *DrawingPage*. Tato třída má funkce, které reagují na interakce vstupního zařízení: myši nebo pera grafického tabletu. Jako výsledek těchto interakcí se vytváří křivka typu *StrokePair*, která se skládá ze seznamu teček a nastavení tužky v momentě kreslení jednotlivé křivky. Každá probíhající křivka je dynamicky vykreslována widgetem *buildCurrentPath* pomocí třídy *SingleLinePainter*, která dědí od třídy *CustomPainter* a slouží k vykreslování grafického obsahu.

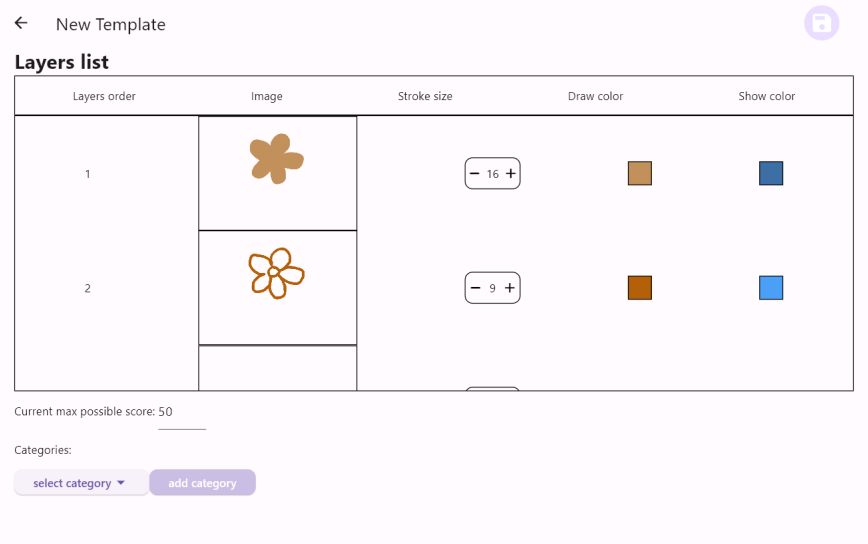
Kromě toho v třídě *SingleLinePainter* není vykreslena pouze křivka, která se skládá jen z původních bodů, ale pro lepší zážitek z používání grafického tabletu tato třída vykresluje křivku s ohledem na hodnotu tlaku grafického pera. K tomu se používá externí knihovna Perfect Freehand [21]. Při kreslení pomocí myši je takové chování simulováno knihovnou, protože myš nemá žádné senzory tlaku.

Jakmile uživatel odstaví vstupní zařízení a ukončí kreslení jednotlivé křivky, aktuální křivka je přidána do seznamu křivek běžné vrstvy. Pro vykreslení celého seznamu křivek aktuální vrstvy se používá widget *buildAllPaths* a třídu *CurrentLayerPainter*. U všech tříd, které vykreslují grafický obsah, je způsob vykreslování křivek stejný jako je popsáno výše u třídy *SingleLinePainter*. Rozdíl spočívá v různých vstupech tříd a zpracování kresby až po vykreslení křivek. Například *CurrentLayerPainter* v procesu překreslení přijímá na vstupu celý seznam křivek a případně se zabývá vykreslením dříve uloženého, rozpracovaného obrazu, pokud existuje.

Pro kreslení jedné vrstvy může uživatel použít pouze jednu barvu, což je nutné pro překreslení šablony. Ve chvíli, kdy uživatel rozhodne o přidání další barvy, musí vytvořit novou vrstvu šablony. Tuto vrstvu přidá tlačítkem a tím pádem je seznam křivek, nakreslených v předchozí vrstvě, přidán do hlavního seznamu, který uchovává křivky všech vrstev. Pro vykreslení předchozích vrstev se používá widget *buildAllLayers* a třída *LayersPainter*. Uživatel je schopen přepínat mezi existujícími vrstvami, přidávat do nich další křivky, ale žádná vrstva nesmí být prázdná na výstupu.

Ve chvíli, kdy uživatel rozhodne dokončit novou šablonu a stiskne tlačítko uložení, jsou nejdůležitější objekty třídy *DrawingPage* předány třídě *SavePainter* a následně *SavePage* (Obrázek 4.3), kde uživatel zvolí poslední nutná nastavení šablony a odešle data na server. Třída *SavePainter* není následníkem *CustomPainter*, nicméně tato třída také zpracovává grafický obsah všech šablon, ale ukládá výstup do jednotlivých souborů, které jsou následně odeslány na server.

Obrázek 4.2: Ukázka kreslení vrstvy nové šablony.



Obrázek 4.3: Uložení nové šablony.

4.2.4 Překreslení podle existující šablony

Pro překreslení se používají třídy *RedrawModePage* a *ColoringPage* (Obrázek 4.4). Jejich základní struktura je stejná jako používání tříd v předchozí sekce. *RedrawModePage* je třídou-předkem, kde se nacházejí nastavení kresby a interaktivní prvky, zatímco *ColoringPage* je třídou-následníkem, která představuje kreslící plátno a zahrnuje logiku překreslení.

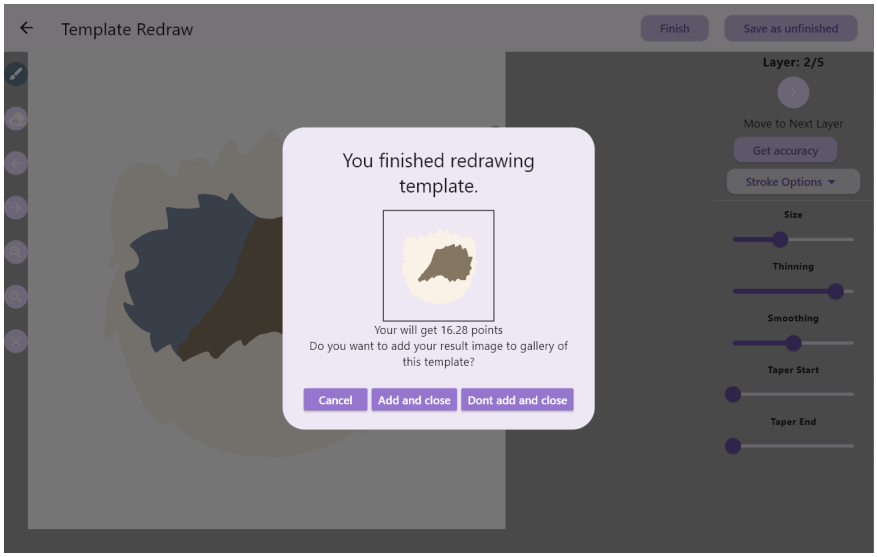
Procesy vykreslování grafického obsahu a zpracování uživatelského vstupu jsou stejné jako v třídě *DrawingPage*, ale *ColoringPage* obsahuje další zajímavé funkce. Uživatel, který překresluje obrázek podle nějaké šablony, musí vidět, jaké prvky má nakreslit. Proto jsou uložené vrstvy původní šablony načítání v *ColoringPage* jako pozadí, podle kterého uživatel rozumí, co a kde přesně má nakreslit. Barva kresby je v daném okamžiku neměnná a je definována autorem šablony. Obrázky pozadí se mění, když uživatel stiskne tlačítko přechodu na další vrstvu, až dokud nedojde ke konci počet vrstev původní šablony. Poté následuje tzv. „vlastní“ vrstva, kde uživatel může přidávat libovolné křivky libovolných barev. Během procesu překreslení není možné přepínat mezi vrstvami. Uživatel má přístupnou pouze aktuální vrstvu, a ve chvíli, kdy provede přechod na další vrstvu, předchozí vrstvu už nemůže změnit.

Pro každou vrstvu je vypočítáno procento přesnosti překreslení vzhledem k originálnímu obrázku, více o tomto procesu je popsáno dále (sekce 4.2.5). Kromě toho v třídě *ColoringPage* probíhá komunikace se serverem, která je nutná k aktualizaci stavu serverového objektu *TemplateUsage*, viz diagram 3.3. Ve chvíli, kdy uživatel dokončí překreslení stisknutím tlačítka *„Dokončit“*, vygeneruje se výsledný obrázek celého překreslení (Obrázek 4.5), který uživatel může uložit na server jako objekt typu *GalleryImage*. Dále se odesílá na server aktuální stav pro dané překreslení (*TemplateUsage*).

Další možností uložení stavu objektu *TemplateUsage* je uložení rozpraco-

vané kresby. V tomto případě se generuje obrázek *finishedLayerAtch*. Obrázek obsahuje křivky, které se vztahují k již nakresleným vrstvám. Dále se generuje obrázek *actualLayerAtch*, který odpovídá obsahu aktuální vrstvy. Oba obrázky jsou uloženy na serveru. Pokud za nějakou dobu uživatel bude pokračovat v uložené rozpracované kresbě, načtou se obrázky *finishedLayerAtch* a *actualLayerAtch*. První bude neměnný, ale druhý obrázek se dá změnit: uživatel může dokreslit křivky nebo smazat existující.

Obrázek 4.4: Překreslení existující šablony.



Obrázek 4.5: Ukončení překreslení šablony.

4.2.5 Výpočet přesnosti překreslení

Pro každou vrstvu v třídě *ColoringPage* se vypočítá procento přesnosti překreslení vzhledem k originálnímu obrázku. Funkce postupně prochází všechny pixely originálního obrázku a obrázku vytvořeného z kresby uživatele a porovnává, zda pixely na stejných pozicích jsou shodné. To se počítá porovnáním průhlednosti jednotlivých pixelů. Také, pokud v uživatelském obrázku nějaký pixel není průhledný a odpovídá pixelu z originálního obrázku, vypočítá se jeho sousední pixely do hloubky 3 a systém je započítává do stejných pixelů. Toto je provedeno proto, aby měl uživatel možnost udělat malé chyby. Podobnou přesnost systém vypočítává na žádost uživatele a také pro každou vrstvu, pak vypočítá střední hodnotu procenta přesnosti a dopočítává výsledné body za překreslení z maximálního možného počtu bodů.

Samozřejmě, algoritmus výpočtu přesnosti momentálně má triviální realizaci. Důvodem je, že během implementace tohoto projektu nebyl výpočet přesnosti stanoven jako prioritní úkol a dané řešení slouží pouze k ověření základního konceptu ohodnocení uživatelské kresby. Navíc nedostatek času ovlivnil rozvoj této funkcionality. V budoucím vývoji je vhodné změnit způsob výpočtu na přesnější řešení.

Kapitola 5

Nasazení

5.1 Nasazení Spring boot aplikace

Pro účely uživatelského testování bylo zvoleno nasazení serverové Spring Boot aplikace na externí host. Jelikož se nejedná o dlouhodobý, plný provoz aplikace, host byl vybrán s ohledem na jednoduchost nasazení a bezplatnost služby. V důsledku toho byl vybrán servis Render.com, který poskytuje neplacené nasazení Spring Boot aplikací. Samozřejmě, vzhledem k neplaceným tarifům má tato služba omezení, například pro tento projekt jsou omezení následující:

- Server se vypíná, pokud během 10 minut nejsou přicházející požadavky. Restart trvá kolem 2-3 minut, jakmile přijde první požadavek, který server „probudí“.
- Server trvale neukládá soubory, všechny uložené soubory jsou k dispozici jenom po dobu běhu serveru. Po vypnutí a restartu je úložiště prázdné.

Proces nasazení je velmi jednoduchý z pohledu programátora. Existuje více možností runtime prostředí aplikace, které se liší různými typy runtime. Pro tento projekt byl použit typ Docker.

Nejprve byl vytvořen .jar soubor. Poté byl vytvořen Dockerfile, jehož Render.com používá k vytvoření Docker image, který slouží k spuštění Java aplikace. Poté bylo na stránkách Render.com zvoleno vytvoření nového webového servisu a připojen GitHub repozitář, ve kterém jsou umístěné všechny soubory aplikace, včetně .jar souboru a Dockerfile. V následujícím kroku byla vybrána potřebná nastavení: název servisu, region hostingu, runtime prostředí a neplacený tarifní plán. Kromě uvedených nastavení je možné také přidat nastavení repozitáře (branch, root directory), proměnné prostředí a další nepovinné možnosti.

Hlavními klady této neplacené služby jsou:

- Připojení GitHub/GitLab repozitáře a automatické nasazení aplikace.
- Automatická reakce na nové commity v repozitáři a okamžité znovunasazení obsahu repozitáře.
- Přístup k logům aplikace v reálném čase.

- Neomezený počet vytváření různých webových servisů.

Render.com má své klady a zápory, ale pro účely uživatelského testování poskytované služby jsou uspokojivé. Před testováním jsou testeři informováni o omezeních, které ovlivňují proces testů, například že první požadavek na server bude trvat 2 minuty, protože server se rozjíždí.

5.2 Nasazení databáze

Spring Boot aplikace používá PostgreSQL databázi a pro její nasazení byl také použit servis Render.com, který poskytuje omezený neplacený hosting PostgreSQL databáze. Omezení zahrnují:

- Kapacita úložiště je maximálně 1 GB.
- Dostupnost služby pouze 90 dnů od data vytvoření.
- Existence pouze jedné PostgreSQL databáze pro jednoho uživatele.
- IP adresa, ze které je nutný externí přístup k databázi, musí být zapsána v nastavení servisu.

Render.com byl vybrán pro nasazení databáze, protože interní spojení databáze a webového servisu je lepší volbou než hostování databáze na separatném servisu.

Vzhledem k omezení běhu samotné Spring Boot aplikace vzniklo omezení databáze, které ovlivňuje proces testování. Jedná se o to, že po vypnutí serveru všechny soubory vytvořené za běhu serveru jsou smazány. Přitom samotná databáze není automaticky vymazána, nevypíná se a je dostupná i když webová služba je vypnutá. V procesu nahrávání souborů na server se v databázi vytváří záznam o jednotlivém souboru, který aplikace dále používá pro získání cesty k souboru. Protože vztahy mezi databází a uloženými soubory nepředpokládají, že soubory mohou být smazány něčím kromě příkazů v aplikaci, Spring Boot aplikace je modifikovaná tak, že po každém restartu jsou smazány všechny záznamy v databázi. Toto omezení se týká pouze nasazené aplikace a je uděláno proto, aby neexistující vztahy nezpůsobily chyby běhu celé aplikace během uživatelského testování.

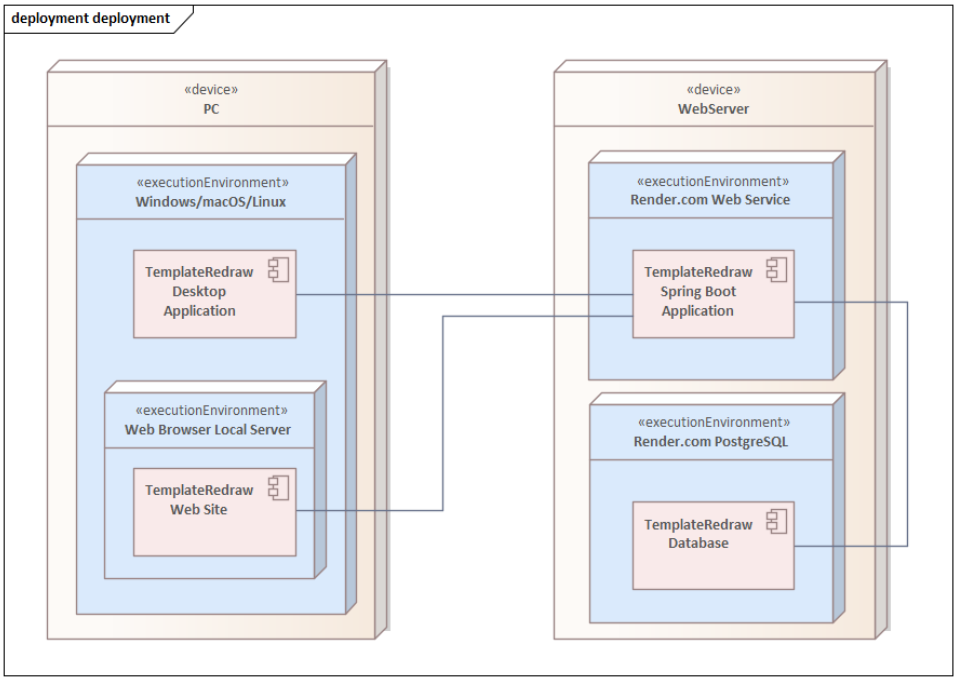
5.3 Klientská aplikace

Pro účely uživatelského testování byly vytvořeny dvě verze desktopové aplikace: pro operační systém Windows a MacOS. Testeři si stáhli a nainstalovali jednu z těchto verzí a používali je k vyzkoušení funkcí.

Vývoj předpokládal vytvoření pouze „počítačových“ verzí aplikace: Flutter poskytuje sestavení aplikace pro počítače, smartphony, prohlížeče, ale v rámci dané práce byla aplikace vyvíjena pouze pro počítače a prohlížeče. Nicméně verze pro prohlížeč nebyla nasazena z důvodu nedostatku času na nasazení.

Momentálně je možné ji spustit pouze na lokálním serveru, například pomocí příkazu *"python -m http.server 8000"*. Desktopová verze potřebuje pouze lokální spuštění na počítači. Celkově jsou webová a desktopová verze stejné: obě potřebují trvalé připojení k síti, používají stejné bezpečnostní mechanismy a mají stejné rozhraní. Verze pro smartphony není použitelná, protože nebylo vytvořeno rozhraní pro takové typy zařízení.

5.4 Schema nasazení



Obrázek 5.1: Diagram nasazení.

Diagram schematicky ukazuje provedené nasazení aplikace.

Kapitola 6
Testování

Po implementaci aplikace bylo provedeno uživatelské testování. Cílem tohoto testování bylo zjistit dostupnost aplikace pro uživatele: pochopitelnost interakce s aplikací, zručnost používání a příjemnost rozhraní. Testování zahrnovalo kognitivní průchody předem definovaných scénářů a také celkové hodnocení aplikace, včetně doporučení uživatelů ohledně jednotlivých prvků nebo funkcí. Celkem se testování zúčastnilo 5 testerů, což je střední optimální počet uživatelů, který může identifikovat až 85% problémů s použitelností aplikace [25]. S ohledem na cílovou skupinu aplikace byla většina testerů vybrána ve věku od 12 do 17 let, ale zároveň několik testerů bylo dospělých.

6.1 Scénáře kognitivního průchodu

Pro účely kognitivního průchodu byly definovány 3 uživatelské scénáře, které zahrnují důležité funkce aplikace. Každý scénář obsahuje konečný cíl testování, jednotlivé akce, které uživatel musí pokusit splnit v definovaném pořadí, a 3 otázky pro každou akci, které jsou definovány pro získání dat o uživatelském zážitku použití aplikace. Otázky jsou označeny v grafu 6.3.1 zkratkami **Q1, Q2, Q3**:

- **Q1** - „Je pro Vás zřejmá správná akce?“
- **Q2** - „Jste schopni spojit danou akci s konečným cílem průchodu?“
- **Q3** - „Dostal jste vhodnou zpětnou vazbu po provedení dané akce?“

Dále jsou popsány jednotlivé scénáře:

6.1.1 Vytvoření vlastní šablony

Cíl scénáře: Vytvořit vlastní šablonu a uložit ji do knihovny šablon. Pořadí potřebných akcí:

1. Přihlásit se/vytvořit účet.
2. Otevřít stránku „Moje knihovna“ („My Library“).
3. Zvolit režim vytvoření šablony, vybrat možnost „kreslení v aplikaci“.

4. V první vrstvě nakreslit křivku červenou barvou a velikosti tužky 20.
5. V druhé vrstvě nakreslit křivku zelenou barvou a velikosti tužky 10.
6. Ukončit kreslení a přejít na další stránku přes tlačítko uložení.
7. Na nové stránce změnit autogenerované barvy pozadí pro každou vrstvu.
8. Přidat šablonu do několika kategorií, pokud s systému existují.
9. Nastavit hodnotu bodů za překreslení na 20 bodů.
10. Uložit šablonu.

6.1.2 Překreslení šablony

Cíl scénáře: Překreslit šablonu a uložit výsledný obrázek do galerie. Pořadí potřebných akcí:

1. Přihlásit se/vytvořit účet.
2. Vybrat si z knihovny šablonu, která obsahuje alespoň 2 vrstvy, otevřít stránku šablony.
3. Začít překreslení šablony.
4. Překreslit všechny existující vrstvy šablony s libovolným procentem přesnosti překreslení.
5. Ve „vlastní“ vrstvě nakreslit křivku libovolné barvy.
6. Ukončit („Finish“) překreslení s přidáním výsledného obrázku do galerie.

6.1.3 Hodnocení šablony

Cíl scénáře: Ohodnotit šablonu a obrázek v galerii. Pořadí potřebných akcí:

1. Přihlásit se/vytvořit účet.
2. Otevřít stránku libovolné šablony obsahující obrázek v galerii.
3. Přidat komentář na stránce šablony.
4. Ohodnotit („Like“/„Dislike“) šablonu.
5. Ověřit stránku libovolného obrázku z galerie.
6. Okomentovat obrázek nebo odpovědět na existující komentář.
7. Nahlásit obrázek s zadáním vlastního důvodu nahlášení.

6.2 Otevřené otázky hodnocení

Kromě kognitivního průchodu přesně definovanými scénáři, testerům byly postaveny otevřené otázky, otevřené otázky, zjištující jejich osobní zážitek a zpětnou vazbu:

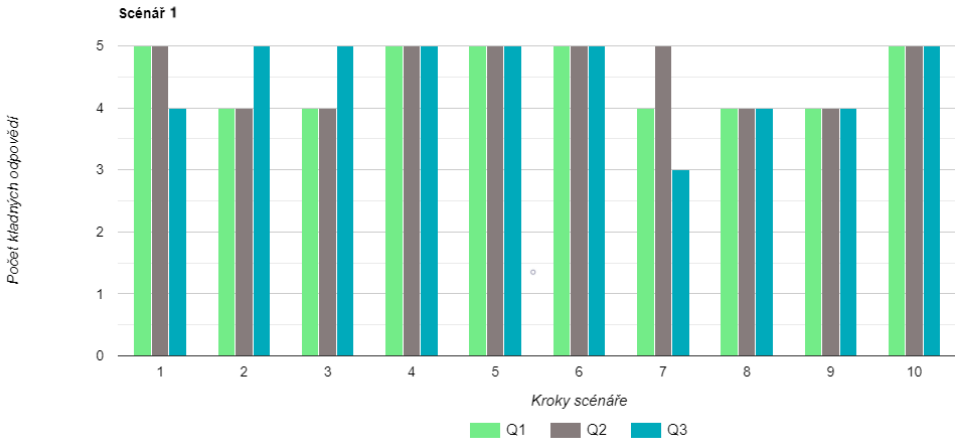
1. Celkové hodnocení aplikace: Je pro Vás zajímavá, srozumitelná? Je pro Vás pochopitelné uživatelské rozhraní?

2. Co byste v aplikaci změnili nebo vylepšili? Popište jednotlivé prvky a funkcionality.

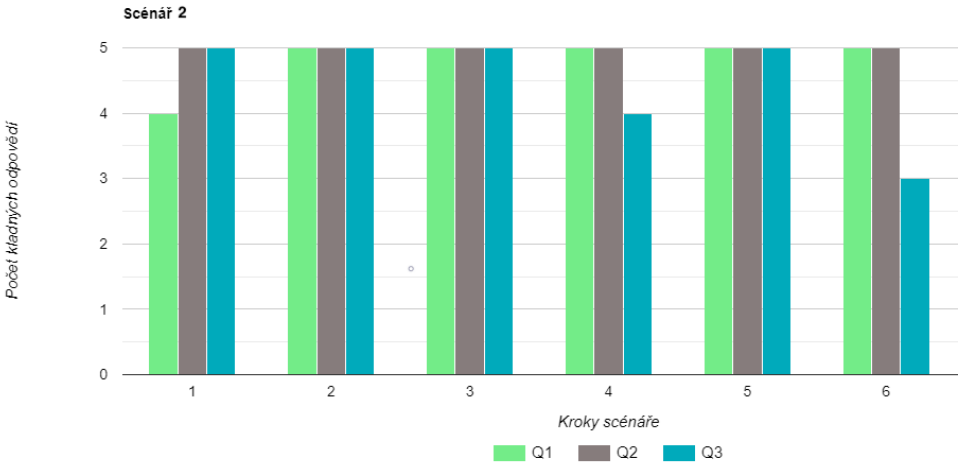
3. Používal byste aplikaci v běžném životě, pokud by aplikace byla plně funkční?

6.3 Výsledky

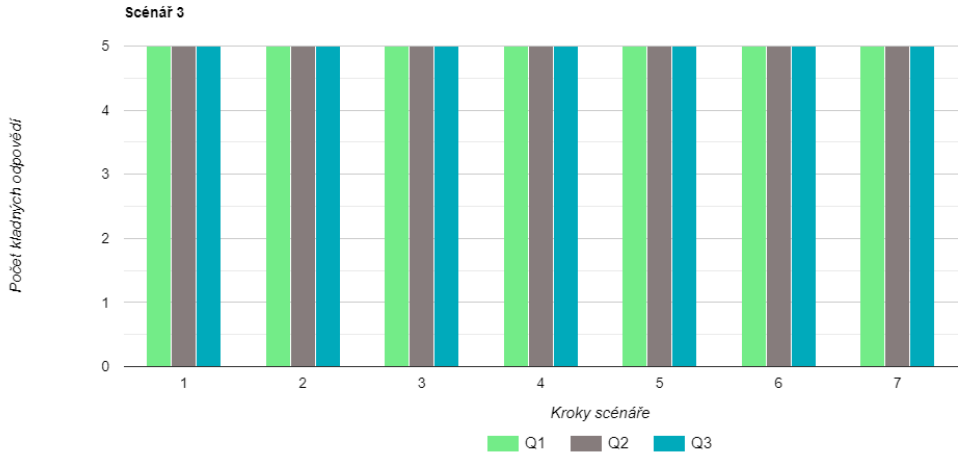
V grafů jsou uvedené získané kladné odpovědi na otázky v jednotlivých scénářích. Každý graf představuje jeden scénář.



Obrázek 6.1: Scénář 1.



Obrázek 6.2: Scénář 2.



Obrázek 6.3: Scénář 3.

Jak je vidět z grafů, většinou jsou jednotlivé akce pro uživatele zřejmé. Funkce, které jsou podobné prvkům známým z jiných aplikací (například třetí scénář, graf 6.3), nezpůsobily uživatelům žádné problémy při testování. Naopak funkce, které jsou unikátnější nebo charakteristické pro kreslící aplikace, nejsou tak zřejmé pro některé testery.

Dále jsou popsány odpovědi na otevřené otázky. Odpovědi testerů na jednu otázku jsou shrnuty do jednoho bloku, který popisuje celkovou zpětnou vazbu.

1. Celkové hodnocení aplikace. Celkově testeři kladně hodnotili aplikaci, větší část rozhraní a funkcí byla pro ně pochopitelná. Pro testery, kteří v běžném životě používají kreslící aplikace, byly funkce týkající se kreslení většinou zřejmé, protože tito uživatelé jsou na podobné akce intuitivně zvyklí. Naopak pro testery, kteří nepoužívají aplikace pro kreslení, bylo někdy obtížnější propojit název akce s výsledkem její interakce, a průchody kreslícími scénáři

vyžadovaly pro takové testery více času. Graf 6.1 dobře ilustruje tuto situaci.

Akce, které jsou pro testery známé z jiných aplikací, které používají každý den, nezpůsobily žádné problémy. Je to vidět i z grafu 6.3.

V otázce designu měli testeři různé názory. Někomu se design líbil a byl příjemný, někomu chyběl tmavý režim aplikace.

2. Prvky a funkce, které by testeři v aplikaci změnili nebo vylepšili. Během testování bylo nalezeno několik chyb, které neovlivnily proces testování, ale byly viditelné pro uživatele. O těchto prvcích testeři informovali autora práce. Co se týče doporučení pro změny nebo vylepšení existujících prvků:

- Zvýraznit světlá tlačítka, aby je uživatelé lépe viděli.
- Přidat klávesové zkratky pro funkce v režimu kreslení, například přepínání mezi tužkou a gumou.
- Přesunout pozici „My Library“ z rozbalovacího menu do hlavičky aplikace.
- Změnit tlačítko „Me“ na avatar uživatele.
- Přidat podporu různých jazykových symbolů.
- Ukazovat seznam všech objektů GalleryImage vytvořených jedním uživatelem v „My Library“.

3. Jestli by testeři používali aplikaci v běžném životě. Testeři uvedli, že by aplikaci používali ve svém životě, ale v různých kontextech. Pro tři uživatele by taková aplikace sloužila jako soutěž s kamarády o získání nejvíce bodů přesnosti za překreslení jednou za nějakou dobu. V takovém případě oni by aplikaci nepoužívali moc často.

Jeden uživatel, který má rád kreslení, uvedl, že by aplikaci používal k sdílení své tvorby ne zcela standardním způsobem. Myslel tím, že by rád vytvářel složité šablony, které by mohly ukázat jeho kreslící schopnosti a zároveň tvořily výzvu pro uživatele, kteří by chtěli takovou šablonu překreslit.

Poslední tester uvedl, že aplikaci by nejspíš nepoužíval, protože nemá rád kreslení. Použil by ji možná jenom na pozvání kamarádů a překreslil by pár šablon. Jako příklad podobné situace z jeho života uvedl hru Gartic Phone [26], kterou hraje jenom občas na pozvání kamarádů.

Kapitola 7
Závěr

Cílem této práce bylo analyzovat, navrhnout a implementovat kolaborativní kreslící aplikaci. Na základě definovaných cílů byla provedena analýza existujících řešení a vyhodnocení jejich výhod a nevýhod. Dále byla provedena rešerše technologií, na jejímž základě bylo rozhodnuto vytvořit multiplatformní aplikaci. Poté následovalo porovnání multiplatformních frameworků pro klientskou aplikace a na základě výsledků byl vybrán framework Flutter. Vzhledem k tomu, že vybraný framework používá programovací jazyk Dart, byly popsány základní prvky tohoto jazyka. Dále bylo provedeno porovnání backendových frameworku a vybrán framework Spring Boot. Na základě popisu aplikace byly definovány funkční a nefunkční požadavky a uživatelské scénáře. S ohledem na požadavky a scénáře byly vytvořeny návrhy obrazovek, které schematicky ukazují uživatelské rozhraní aplikace. Návrhy byly připraveny pro desktopovou verzi aplikace.

Další část práce je věnována návrhu aplikace. Byl vytvořen datový model a sekvenční diagramy. Sekvenční diagramy byly vytvořeny pro popis komunikace mezi lifeliny pro vybrané uživatelské scénáře.

V implementační části byla popsána struktura realizované aplikace a několik netriviálních funkcí. Byla představena serverová část aplikace, vytvořená ve frameworku Spring Boot, a klientská část, vytvořená ve frameworku Flutter. Obě části aplikace byly vyvíjeny paralelně, což umožnilo průběžné vývojářské testování jednotlivých funkcí.

V kapitole Nasazení byly popsány vybrané technologie pro nasazení systému a nezbytná omezení, která vznikla kvůli nasazení.

Následovala kapitola Testování. Během této fáze byli vybráni testeři mezi dětmi a mládeží a byly provedeny kognitivní průchody definovanými scénáři. Získaná zpětná vazba od testerů byla vzata v úvahu pro budoucí pokračování v implementaci aplikace.

Všechny cíle, které tato práce kladla, jsou považovány za splněné. Vývoj aplikace bude pokračovat s ohledem na výsledky uživatelského testování.

7.1 Vyhodnocení provedené práce

Během implementace bylo stanoveno za cíl implementovat nejdůležitější části návrhu aplikace, popsané v zadání této práce. Nicméně, podařilo se

implementovat některé prvky rozšiřující základ aplikace. Většina těchto funkcí je implementována pro ověření a otestování myšlenky návrhu a pro lepší uživatelský zážitek během fází testování.

Základ aplikace zahrnoval dle zadání implementaci následujících částí:

- proces vytvoření šablony přes kreslení v aplikaci
- proces překreslení šablony
- knihovnu vytvořených šablon, přes kterou uživatel je schopen začít proces překreslení
- spojené s hlavními body nutné prvky, například uživatelské účty včetně rozdělení práv dle rolí, kategorie šablon, ukázkové obrázky pro každou šablonu

Po implementaci hlavních bodů byly implementovány následující prvky nad rámec původního rozsahu:

- hodnocení šablon a obrázků v galerii, které zahrnuje přidání značek „Like“ nebo „Dislike“ a napsání komentáře
- nahlášení šablon, obrázků a uživatelských účtů a spojené s tím spravování nahlášení administrátory a moderátory
- vytvoření šablony přes nahrání předem připravených PNG souborů
- výpočet bodů přesnosti překreslení šablony
- uložení rozpracované kresby v procesu překreslení
- přidání do galerie obrázků, které vznikly z procesu překreslení šablony

7.2 Výhled do budoucna

Z důvodu nedostatku času nebo nižší priority některých částí aplikace nebyly implementovány následující funkce, které ale nejsou obsahem původního zadáni:

- uživatelské rozhraní pro smartphony
- globální vyhledávání
- proces verifikace šablon a s tím spojené rozdělení uživatelských ratingů na celkový a globální. Momentálně se body za veškeré překreslení započítávají do globálního ratingu uživatele
- blokace uživatelů a šablon
- vylepšení některých funkcí, které obsahují jenom základní implementace myšlenky, například výpočet bodů přesnosti, nastavení vrstev
- vylepšená validace uživatelské interakce

Budoucí implementace aplikace bude zahrnovat realizaci uvedených funkcí a jejich následné testování.

Příloha A

Literatura

[1] *Aggie.io, webová aplikace, verze 1.5.14.* (2022) [online]. [cit. 2024-05-12]. Dostupné z: `https://aggie.io/`

[2] Jonathan Knispel, Fraser Bullock. *Collaborative VR painting in web browsers.* (2017) [online]. [cit. 2024-05-12]. Dostupné z: `https://dl.acm.org/doi/abs/10.1145/3139468.3148451`

[3] *ArtWorkout, iPad aplikace, verze 1.4.11.* (2023) [online]. [cit. 2024-05-12]. Dostupné z: `https://apps.apple.com/us/app/artworkout-learn-how-to-draw/id1564657118`

[4] Ben Bederson, Allison Druin, Lisa Sherman. *Designing a Collaborative Finger Painting Application for Children.* (2000) [online]. [cit. 2024-05-12]. Dostupné z: `https://www.researchgate.net/publication/2815298_Designing_a_Collaborative_Finger_Painting_Application_for_Children`

[5] *Coloring Book for Me, IOS aplikace, verze 4.48.4.* (2023) [online]. [cit. 2024-05-12]. Dostupné z: `https://apps.apple.com/us/app/coloring-book-for-me/id1093108529?platform=iphone`

[6] *JPen, Java knihovna, verze 2-150301.* (2015) [online]. [cit. 2024-05-13]. Dostupné z: `https://jpen.sourceforge.net/html-home/index.html`

[7] Jessica David, Brian Eoff, Dr. Tracy Hammond. *CoSke-An Exploration in Collaborative Sketching.* (2010) [online]. [cit. 2024-05-13]. Dostupné z: `https://www.researchgate.net/publication/228845129_CoSke-An_Exploration_in_Collaborative_Sketching`

[8] *Perfect-Freehand, Typescript knihovna, verze 1.2.0.* (2022) [online]. [cit. 2024-05-13]. Dostupné z: `https://www.npmjs.com/package/perfect-freehand/v/1.2.0?activeTab=readme`

[9] *Electron, JavaScript framework.* [online]. [cit. 2024-05-13]. Dostupné z: https://www.electronjs.org/

[10] Lionel Sujay Vailshery. *Cross-platform mobile frameworks used by software developers worldwide from 2019 to 2022.* (2023) [online]. [cit. 2024-05-13]. Dostupné z: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/

[11] Marco L. Napoli. *Introducing Flutter and Getting Started.* (2019) [online]. [cit. 2024-05-13]. Dostupné z: https://www.researchgate.net/publication/335849904_Introducing_Flutter_and_Getting_Started

[12] Art Yudin. *Introduction to React Native.* (2020) [online]. [cit. 2024-05-13]. Dostupné z: https://www.researchgate.net/publication/347626725_Introduction_to_React_Native

[13] U Urathal Alias Sri Swathiga, P.Vinodhini, V.Sasikala. *An interpretation of Dart programming language.* (2022) [online]. [cit. 2024-05-13]. Dostupné z: https://www.researchgate.net/publication/358661479_AN_INTERPRETATION_OF_DART_PROGRAMMING_LANGUAGE

[14] Geetha S, Devang Dalvi, Dr. Mahesh Tandel. *Bootstrap and Django Framework.* (2021) [online]. [cit. 2024-05-15]. Dostupné z: https://www.researchgate.net/publication/357073883_Bootstrap_and_Django_Framework

[15] Sanjib Sinha. *Beginning Laravel, Build Websites with Laravel 5.8.* (2019) [online]. [cit. 2024-05-15]. Dostupné z: https://www.researchgate.net/publication/335668475_Beginning_Laravel_Build_Websites_with_Laravel_58

[16] Jakub Suchanowski. *Performance analysis of web applications created in the Spring and Laravel frameworks.* (2023) [online]. [cit. 2024-05-15]. Dostupné z: https://www.researchgate.net/publication/376970887_Performance_analysis_of_web_applications_created_in_the_Spring_and_Laravel_frameworks

[17] Azat Mardan. *Starting with Express.js.* (2014) [online]. [cit. 2024-05-15]. Dostupné z: https://www.researchgate.net/publication/312859271_Starting_with_Expressjs

[18] Binildas Christudas. *Spring Boot.* (2019) [online]. [cit. 2024-05-15]. Dostupné z: https://www.researchgate.net/publication/334025156_Spring_Boot

[19] *REST API vs. GraphQL, ByteByteGo.* [online]. [cit. 2024-05-15]. Dostupné z: https://github.com/ByteByteGoHq/system-design-101?tab=readme-ov-file#rest-api-vs-graphql

[20] *Flutter Hooks, Flutter knihovna.* [online]. [cit. 2024-05-18]. Dostupné z: https://pub.dev/packages/flutter_hooks

[21] *Perfect freehand, Flutter knihovna, github repozitář.* [online]. [cit. 2024-05-18]. Dostupné z: https://github.com/steveruizok/perfect-freehand

[22] *Flutter colorpicker, Flutter knihovna.* [online]. [cit. 2024-05-18]. Dostupné z: https://pub.dev/packages/flutter_colorpicker

[23] *Dio, Flutter knihovna.* [online]. [cit. 2024-05-18]. Dostupné z: https://pub.dev/packages/dio

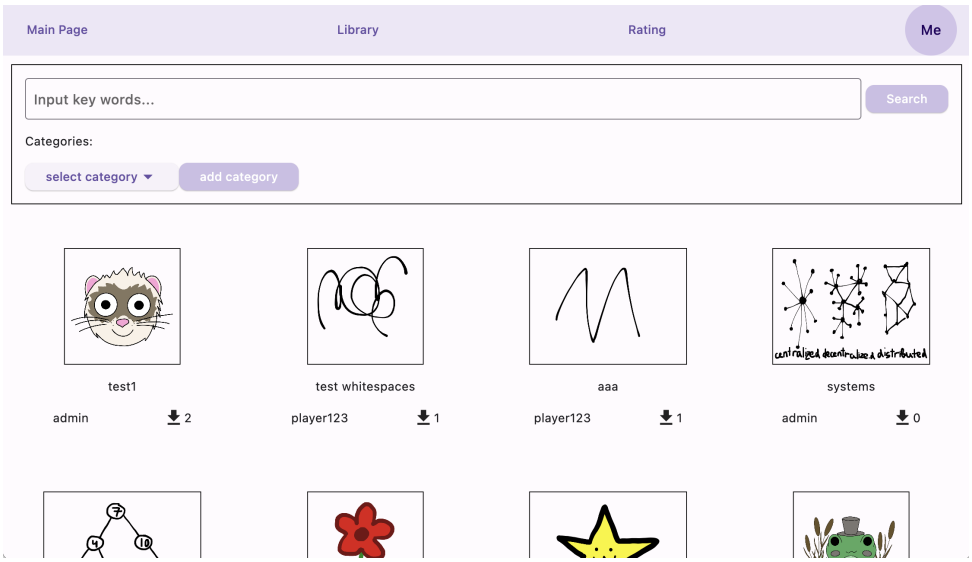
[24] *Cyclop, Flutter knihovna.* [online]. [cit. 2024-05-18]. Dostupné z: https://pub.dev/packages/cyclop

[25] Jakob Nielsen. *Why You Only Need to Test with 5 Users.* (2000) [online]. [cit. 2024-05-18]. Dostupné z: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/

[26] *Gartic Phone Game.* [online]. [cit. 2024-05-18]. Dostupné z: https://garticphone.com/en

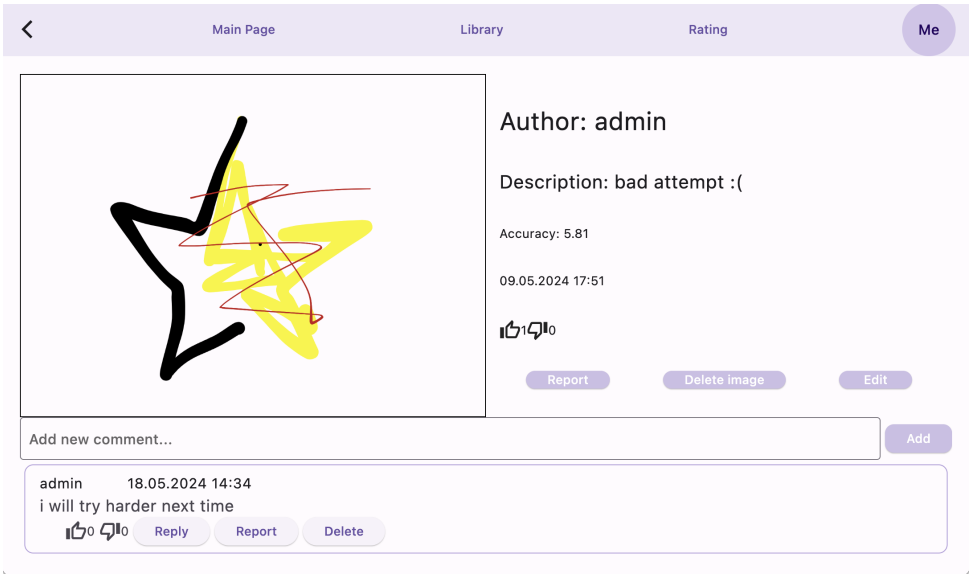
Příloha B

Ukázky obrazovek aplikace

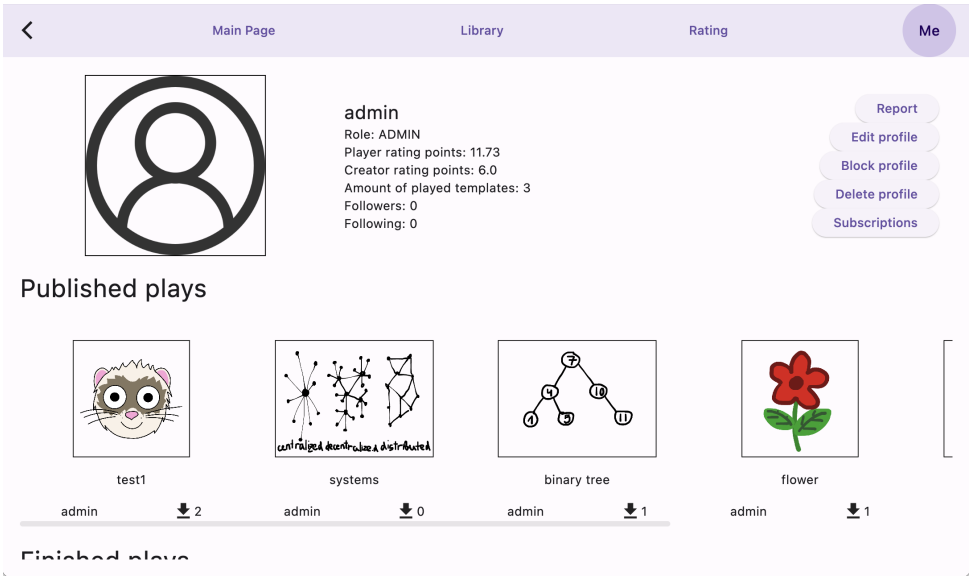


Obrázek B.1: Veřejná knihovna.

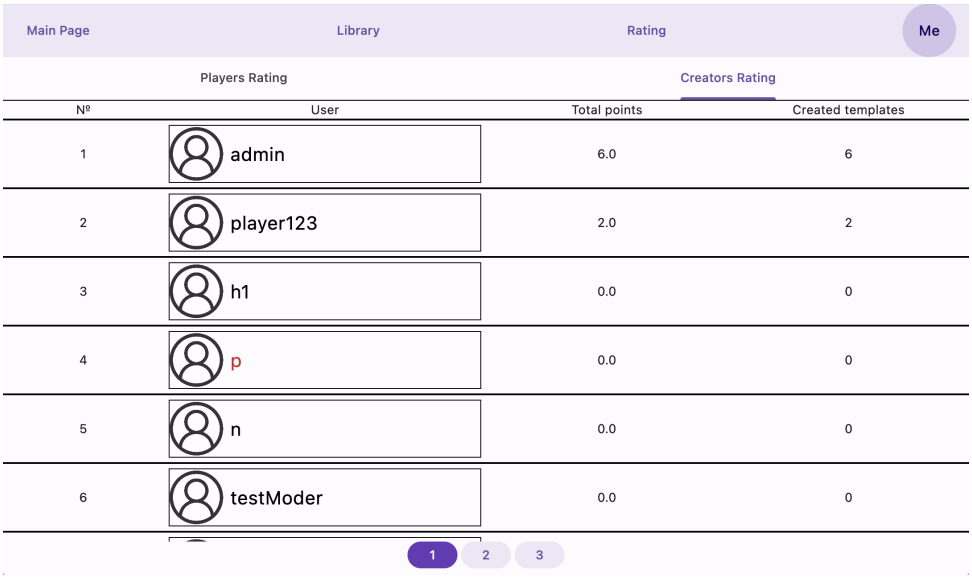
Obrázek B.2: Stránka šablony.



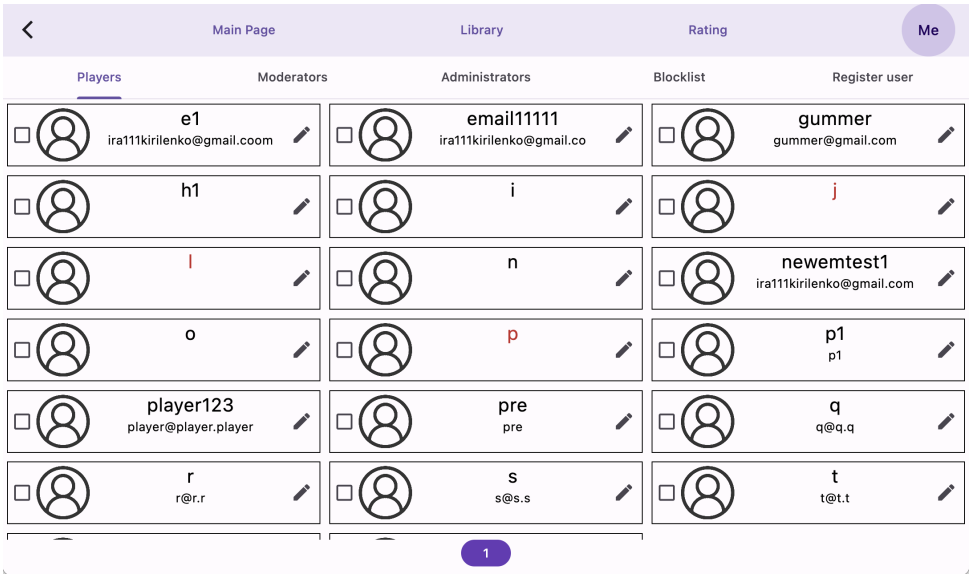
Obrázek B.3: Stránka obrázku z galerie.



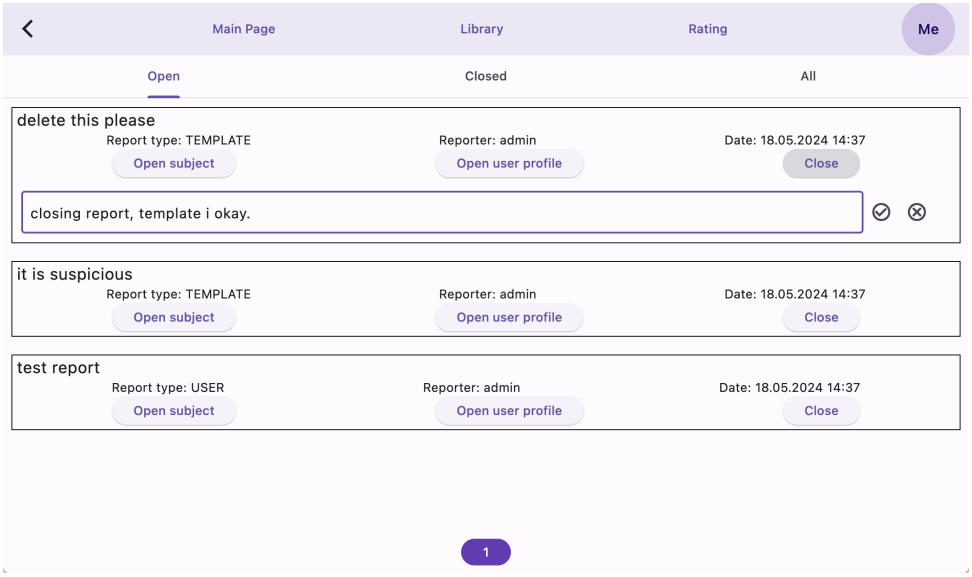
Obrázek B.4: Uživatelský účet.



Obrázek B.5: Seznam uživatelských ratingů.



Obrázek B.6: Seznam uživatelů, který vidí administrator.



Obrázek B.7: Seznam nahlašení, který vidí administrator.