FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Systém pro sledování osobního rozvoje
Student:	Quang Vu Tran
Vedoucí:	Ing. Marek Suchánek
Studijní program:	Informatika
Obor / specializace:	Webové a softwarové inženýrství, zaměření Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Osobní rozvoj je důležitou součástí lidského života a jeho sledování působí motivačně a umožňuje různé analýzy. Existuje řada aplikací zaměřených na specifické aktivity, kterých je mnoho druhů. Cílem práce je na základě analýzy a rešerše navrhnout a implementovat flexibilní systém pro sledování osobních aktivit se zaměřením na osobní rozvoj s prvky gamifikace:

- Analyzujte doménu sledování osobních aktivity, jejich vyhodnocování a další relevantní souvislosti s využitím konceptuálního modelování.
- Proveďte stručnou rešerši existujících řešení.
- Sestavte požadavky a případy užití na základě analýzy a rešerše.
- Navrhněte vlastní řešení formou webové aplikace s responzivním designem. Při návrhu zohledněte možnosti rozšíření, flexibilitu a interoperabilitu.
- Implementujte klíčové požadavky s využitím frameworku Django. Systém musí uživateli umožňovat definovat vlastní aktivity, alespoň se třemi druhy sledovaných hodnot.
- Zhodnoťte přínosy vlastního řešení a navrhněte další rozvoj.

Elektronicky schválil/a Ing. Michal Valenta, Ph.D. dne 30. června 2021 v Praze.

Bakalářská práce

SYSTÉM PRO SLEDOVÁNÍ OSOBNÍHO ROZVOJE

Tran Quang Vu

Fakulta informačních technologií
Katedra softwarového inženýřství
Vedoucí: Ing. Marek Suchánek
4. května 2022

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především Ing. Marku Suchánkovi za vedení mé práce. Během vypracování této práce mi poskytl cenné informace, rady a svůj čas. Také bych chtěl poděkovat mé rodině za podporu během studia a kamarádům, kteří mi poskytovali mentální podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 4. května 2022 .....

Abstrakt

Cílem této bakalářské práce je provést analýzu, návrh a implementaci webové aplikace s využitím frameworku Django a postupů softwarového inženýrství. Aplikace podporuje osobní rozvoj, konkrétně vytvoření nebo zbavení se zvyků. Hlavním stavebním kamenem této aplikace je metoda habit tracking, která je obohacena prvky gamifikace a statistikami. Součástí této práce je také testování použitelnosti, vyhodnocení výsledné aplikace a návrhy dalšího rozvoje.

Klíčová slova webová aplikace, responsivita, osobní rozvoj, zvyky, habit tracking, gamifikace, Bootstrap, Django

Abstract

The aim of this bachelor thesis is to analyze, design and implement a web application using Django framework and software engineering practices. The application supports self-developement, specifically creation or elimination of habits. The main building block of this application is a method called habit tracking, which is enriched with gamification elements and statistics. This thesis also includes usability testing, evaluation of the final application and suggestions for further developement.

Keywords web application, responsivity, self-developement, habits, habit tracking, gamification, Bootstrap, Django

Seznam zkratek

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
CSS	Cascading Style Sheets
DTL	Django template language
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MVT	Model-View-Template
ORM	Object–relational mapping
SQL	Structured Query Language
UI	User Interface
URL	Uniform Resource Locator
XML	Extensible Markup Language

Úvod

Člověk je tvořen svými zvyky a jeho každodenními aktivitami. S rychle se rozvíjecím světem, přichází mnoho kladů i záporů. Některé společnosti poskytující moderní formy zábavy používají lidskou psychiku, aby podvědomě donutily uživatele k nepřirozenému počtu hodin strávených na těchto aktivitách, což může vést k vytvoření zvyklostí, které jsou časově náročné a neproduktivní.

Vytvoření neprospěšných zvyků je značně jednodušší než se jich zbavit. Po uvědomění si tohoto problému jsem se začal o tématiku seberozvoje více zajímat, konkrétněji jak si udržet positivní aktivity/zvyklosti a jak se postupně zbavit těch negativních.

Jednou z metod je habit tracking, která spočívá ve sledování pokroku dané aktivity tím, že si člověk označí dny, ve kterých aktivitu splnil a ve kterých ji nesplnil. Vizualizace pokroku působí motivačně a zároveň může odhalit podvědomé trendy našeho chování. Tato metoda je hlavním stavebním kamenem této práce a současných softwarových produktů.

Téma práce jsem navrhl, protože z existujících řešení, kterých je sice mnoho, jsem nenašel řešení takové, které by mi zcela vyhovovalo. Existující řešení byla buď komplikovaná, hodně zaměřená na data a statistiky, bez zábavných herních prvků anebo příliš jednoduché.

Cílem této práce je provést analýzu domény a existujících řešení, sestavit funkční a nefunkční požadavky, navrhnout a implementovat responsivní webovou aplikaci ve frameworku Django, provést testování použitelnosti uživateli a vyhodnotit přínosy výsledné aplikace a návrh dalšího rozvoje.

Systém umožní uživatelům kompletní autentizaci, správu uživatelského profilu, možnost definovat vlastní aktivity s alespoň třemi druhy sledovaných hodnot, zobrazit přehled stavů jednotlivých aktivit a poskytovat různé statistiky k daným aktivitám. Systém bude provázán prvky gamifikace jako jsou například: úroveň uživatele, milníky pro jednotlivé aktivity apod.

Kapitola 1
Analýza

Analýza je jednou z prvních fází při vývoji softwarového produktu. Je důležité se seznámit s doménou, pro kterou se produkt vyvíjí. Následně seznámit se s existujícími řešeními, zhodnotit jejich klady a zápory, abychom nevytvářeli softwarové řešení problematiky, které by bylo podobné existujícímu řešení a vyvarovat se záporům těchto řešení.
V této kapitole nejprve zanalyzuji sledovanou doménu(zvyky, habit tracking, gamifikace) a zaměřím se na analýzu existujících řešení. V rámci analýzy existujících řešení proberu 3 aplikace a popíšu cíle, ke kterým by měla výsledná aplikace směrovat. V neposlední řadě sestavím funkční požadavky, nefunkční požadavky a případy užití aplikace.

1.1 Analýza domény

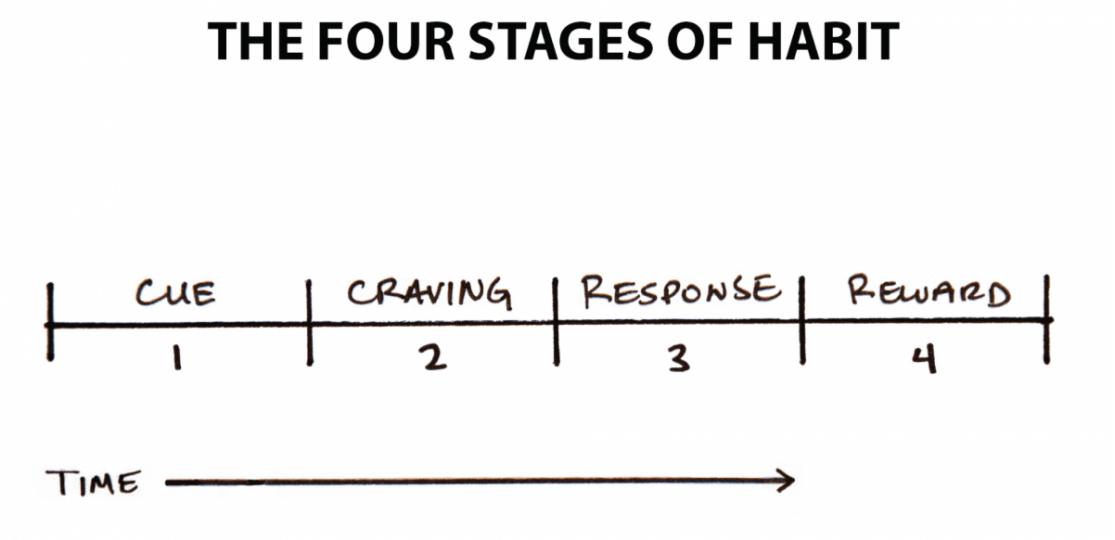
Pro pochopení domény je nutné si definovat zvyk a jak vzniká. Následně se můžeme přesunout k metodě habit tracking, která slouží jako podpůrný systém k vytváření nebo omezení zvyků. Nakonec se podíváme na prvky gamifikace a jejich roli ve zvýšení úspěšnosti k modifikaci naších zvyklostí. Jako zdroj informací jsem využil práce autora James Clear, který napsal světoznámý best-seller Atomic Habits. [1]

1.1.1 Zvyk

Zvyk lze definovat jako obvyklý způsob chování nebo myšlení. Často jde o aktivity, které člověk opakuje pravidelně. Jinými slovy, zvyk je typ chování, které bylo opakováno tolikrát, že se stalo automatickým. Hlavním cílem návyků je řešit problémy s nejméně vynaloženým úsilím a energií.

Jak si vytvořit nové zvyky

Jakýkoli zvyk lze rozdělit do opakujících se akcí sekvence, která zahrnuje čtyři fáze: podnět(cue), touha(craving), reakce(response) a odměna(reward).



Obrázek 1.1 All habits proceed through four stages in the same order: cue, craving, response, and reward. [2]

Podnět dává signál, zahájení potřebného chování k dosažení odměny. Je to informace, která předpovídá odměnu. Lidská mysl neustále analyzuje vnitřní a vnější prostředí, aby nalezla odměny. Protože podnět je první známkou toho, že jsme blízko odměny, přirozeně vede k touze.
Touha je motivační silou každého zvyku. Bez určité úrovně motivace nebo touhy po změně, nemáme důvod jednat. Člověk netouží po zvyku, ale spíše po změně vnitřního stavu, kterou přináší.
Reakce je už přímo zvyk nebo akce, kterou člověk vykonává k získání odměny. Zda k reakci dojde, závisí na tom, jestli člověk je dostatečně motivován a jestli má potřebné schopnosti.
Odměny se dosáhne pokud byla reakce úspěšná.

Podnět je o tom, že si člověk všimne odměny. Touha je o tom, že chce získat odměnu. Reakce je o akcích vedoucí k získání odměny. Dosažení odměny slouží dvěma účelům: uspokojení a získání nových informací. [1]

Mezi některé příklady formulace zvyků patří:

1. **Podnět:** Zazvoní nám telefon s novou zprávou.
2. **Touha:** Chceme zjistit obsah zprávy.
3. **Reakce:** Vezmeme telefon a přečteme si zprávu.
4. **Odměna:** Uspokojili jsme touhu přečíst zprávu. Zazvonění telefonu si spojíme s uchopením telefonu a přečtením nové zprávy.

1. **Podnět:** Učíme se nové těžké téma.
2. **Touha:** Začínáme cítit stres a jsme zahlceni novými informacemi. Chceme se zbavit stresu.
3. **Reakce:** Otevřeme si sociální média.
4. **Odměna:** Uspokojili jsme touhu ztlumit stres. Stres při učení si spojíme s prohlížením sociálních médií.

Jednou z metod pro vytvoření nové positivní aktivity je *Four Laws of Behavior Change*. Tato metoda je založena na pochopení cyklu podnět → touha → reakce → odměna. Při plánování nové aktivity je doporučeno zahrnout následující pravidla:

1. **The 1st law(Cue):** Make it obvious.
2. **The 2nd law(Craving):** Make it attractive.
3. **The 3rd law(Response):** Make it easy.
4. **The 4th law(Reward):** Make it satisfying.

Pro zbavení se špatného zvyku platí inverze:

1. **Inversion of the 1st law(Cue):** Make it invisible.
2. **Inversion of the 2nd law(Craving):** Make it unattractive.
3. **Inversion of the 3rd law(Response):** Make it difficult.
4. **Inversion of the 4th law(Reward):** Make it unsatisfying.

Proč je těžké si udržet positivní zvyky a zbavit se negativních

Mnoho lidí začíná proces změny svých návyků tím, že se zaměří na to, čeho chtějí dosáhnout. To je vede k návykům založených na výsledcích. Alternativou je vybudovat návyky založené na identitě.

Představme si dvě osoby, které se snaží přestat kouřit. Když je první osobě nabídnuta cigareta, řekne: „Ne, děkuji. Snažím se skončit." Zní to jako rozumná odpověď, ale tato osoba stále věří, že je kuřák, který se snaží změnit. Doufá, že se jejich kuřácké zvyklosti změní, zatímco stále nesou stejná přesvědčení o sobě.

Druhá osoba odmítne slovy: „Ne, děkuji. Nejsem kuřák." Je to malý rozdíl, ale toto už signalizuje posun v identitě. Kouření bylo součástí minulosti, ne současnosti.

Většina lidí neuvažuje o změně identity, když se chtějí zlepšit. Stanovují si cíle a určují kroky, které by měli podniknout, aby těchto cílů dosáhli, aniž by zvažovali přesvědčení, která řídí jejich jednání. Neuvědomují si, že jejich stará identita může sabotovat jejich nové plány. To je důvod, proč nemůžeme být připoutaní k jedné verzi naší identity. Pokrok vyžaduje odnaučení přesvědčeních o sebe sama. Chceme-li se stát nejlepší verzí sebe, musíme neustále upravovat svá přesvědčení a rozšiřovat svou identitu. [1]

Příklady zaměření se na cíle oproti identitě:

- Cílem není přestat kouřit, ale cílem by mělo být stát se nekuřákem.
- Cílem není přečíst knihu, ale cílem by mělo být stát se čtenářem.
- Cílem není uběhnout maraton, ale cílem by mělo být stát se běžcem.
- Cílem není naučit se hudební nástroj, ale cílem by mělo být stát se hudebníkem.

Nejspolehlivější formou vnitřní motivace je, když se zvyk stane součástí naší identity. Čím více jsme hrdí na určitý aspekt naší identity, tím více budeme motivováni udržovat návyky s tím spojené. Například pokud je někdo hrdý na to, jak jeho vlasy vypadají, vytvoří si nejrůznější návyky, jak o ně pečovat a udržovat je.

Využití v softwarovém řešení

Metoda *Four Laws of Behavior Change* by mohla být využita v softwarovém řešení, které by bylo zaměřené na vytváření nových zvyků. Podnět pro novou aktivitu může být samotná aplikace. Proto by jako jednou z funkcionalit mohla být možnost, aby uživatelem definované aktivity v daný den byly permanentně zobrazeny, například ve formě horní lišty. Uživatelé by měli rychlý přístup k použití nejdůležitější funkcionality, která by byla neustále na očích.

1.1.2 Habit tracking

Habit tracking je jednoduchý způsob, jak změřit, zda jste udělali aktivitu. Nejzákladnějším formátem je pořídit si kalendář a odškrtávat si každý den, kdy danou aktivitu splníte. [3]

Mezi hlavní benefity patří:

- Jednoduchost metody.
- Připomenutí ke splnění aktivity.
- Vizualizace pokroku.
- Motivace k pokračování.

Využití v softwarovém řešení

Jelikož metoda habit tracking poskytuje přehledné znázornění dat, přirozeně je tato metoda vhodná pro zpracování v softwarovém řešení. Typické softwarové zpracování metody může být klasická „kalendářní“ aplikace, ve které si bude uživatel označovat splnění či nesplnění aktivity.

Jeden z dalších benefitů jednoduchosti reprezentace splněné/nesplněné aktivity je snadnost vizualizovat tyto data ve formátu statistik a grafů. Poskytnutí přehledných statistik uživateli, může přispět k identifikaci trendů našeho chování a v určitých případech může přispět i k udržení motivace(např. uživatel si chce udržet 75% úspěšnost a výše).

Pouhé udržování přehledu splněnosti aktivit je základem. To ale znamená, že je tu prostor pro zpestření metody habit tracking. Jednou z těchto zpestření může být například možnost sledovat jiné vlastnosti daných aktivit jako jsou například:

- Sledování obtížnosti začít danou aktivitu.
- Sledování obtížnosti splnit aktivitu.
- Sledování doby ke splnění aktivity.

Předem zmíněné rozšíření lze snadno implementovat v softwarovém řešení. Zobrazení statistik je práce s existujícími daty a jejich správné zpracování do relevantních statistik. Rozšíření možností sledovaných aktivit lze docílit přidáním nového sloupce v databázové tabulce reprezentující aktivitu nebo nových atributů třídy reprezentující aktivitu apod.

1.1.3 Gamifikace

Gamifikace je aplikace prvků herního designu a principů v neherních kontextech. Lze jej také definovat jako soubor činností a procesů k řešení problémů s využitím vlastností herních prvků. [4]

Prvky gamifikace pomáhají s motivací a dokážou přidat zábavné prvky aktivitám, které jsou bez nich nezábavné a nudné. Mnoho firem začalo implementovat prvky gamifikace, aby si udržely zaměstnance a zabránily jejich vyhoření.

Mezi některé klasické herní prvky patří: úroveň hráče, body, milníky, odměny, žebříčky.

Využití v softwarovém řešení

Jelikož gamifikace je aplikace prvků herního designu a principů, přirozeně je tato metoda hojně využívaná ve video hrách. Programátoři používají různé techniky a triky, aby si udrželi co nejvíce hráčů a vydělali více peněz. Vedle základních vlastností her(hratelnost, grafika, ovládání apod.) hrají prvky gamifikace spíše retenční roli a jsou vedlejšími funkcionalitami.

V dnešní době se prvky gamifikace začaly používat i v softwarových aplikací neherního typu. Zejména v aplikacích zaměřující se na témata vyžadující lidskou vůli(seberozvoj, produktivita apod.). Gamifikace uživatelům těchto aplikací pomáhá k udržení motivace a přidává zábavné herní prvky k úkolům, které by normálně potřebovaly lidskou vůli ke splnění.

Pro implementaci úrovní uživatele je nutné mít systém zkušeností a po dosažení kapacity zkušeností současné úrovně se uživateli zvýší úroveň. Uživatelé by dostávali určitý počet zkušeností splněním různých úkolů, milníků, aktivit apod.

Alternativou k dostávání zkušeností jsou body nebo systém mincí. Uživatelé by dostávali určitý počet splněním různých úkolů, milníků, aktivit apod. Za nastřádané body/mince by si uživatelé mohli koupit různé předměty.

Ve hrách jsou milníky většinou generovány předem nebo automaticky, čili hráč nemá kontrolu nad vytvářením milníků.

Odměny lze rozdat uživatelům po úspěšném ukončení nějakého úkolu, splnění milníku, dosažení nové úrovně, apod. Uživatelům by měl být také zpřístupněn inventář nebo jiný systém ve kterém se uloží odměny a uživatel si je může prohlédnout.

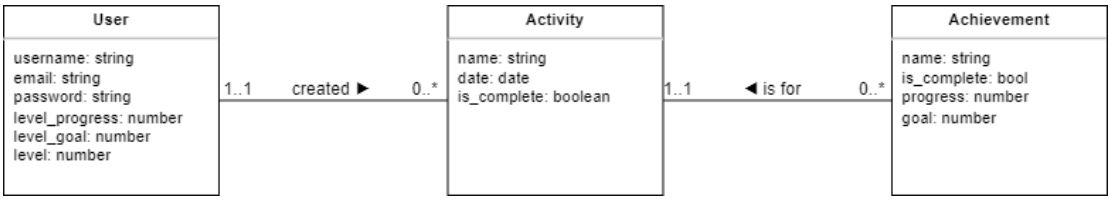
Pokud je potřeba vzplanout soutěživost uživatelů, lze využít různých žebříčků, kde by byly vypsání uživatelé vzestupně seřazený podle jejich dosažené úrovně, různých bodů, splněných úkolů, apod.

Z předem zmíněných prvků jsem se rozhodl pro implementaci systému úrovní a milníků. Jedním z nápadů bylo, že by uživatel dostával zkušenosti každým přihlášením, ale po zvážení praktičnosti jsem se rozhodl, že uživatel bude dostávat zkušenosti každou splněnou aktivitou. Milníky k jednotlivým aktivitám se vygenerují poté, co uživatel vytvoří aktivitu.

1.1.4 Doménový diagram

Doménové modelování je způsob, jak modelovat entity reálného světa a vztahy mezi nimi, které souhrnně popisují prostor problémové domény. [5]

Z poznatků analýzy domény lze navrhnout základní doménový model pro bližší pochopení a znázornění souvislostí jednotlivých částí studované domény.



Obrázek 1.2 Doménový diagram

Uživatel

Entita User představuje uživatele aplikace. Pro bližší identifikaci uživatele se používá přihlašovací jméno a emailová adresa. V entitě se také ukládá současná úroveň uživatele a pokrok do další úrovně.

Aktivita

Entita Activity představuje aktivitu, kterou vytvořil uživatel. Každá aktivita bude mít název, datum aktivity a stav splněno/nesplněno.

Milník

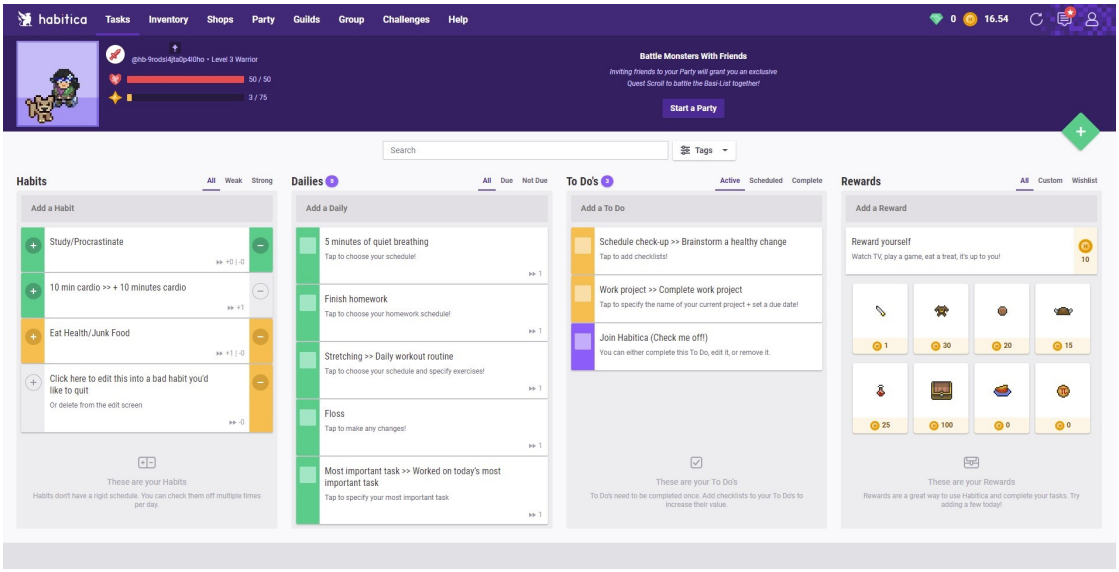
Entita Achievement představuje milník, který je spojen s aktivitou. Každý milník bude mít název, pokrok ke splnění milníku a stav splněno/nesplněno.

1.2 Analýza existujících řešení

Na trhu existuje mnoho řešení, nicméně většina nejpopulárnějších aplikací jsou podporována pouze na platformě Android nebo iOS. Následující aplikace jsem vybral na základě popularity těchto aplikací a jestli jsou podporované jako webová aplikace. Každou aplikaci zhodnotím a popíšu jejich hlavní funkcionality.

1.2.1 Habitica

Habitica je jednou z nejpopulárnějších aplikací, co se týče habit trackingu, protože je silně zaměřená na prvky gamifikace. Kromě základních funkcí pro habit tracking, uživatel dostává mince za splnění aktivit, které si sám může nastavit. Mince může využít pro koupení různých prvků(brnění, účesy, zbraně, mazlíčci apod.) pro modifikaci svého avataru.



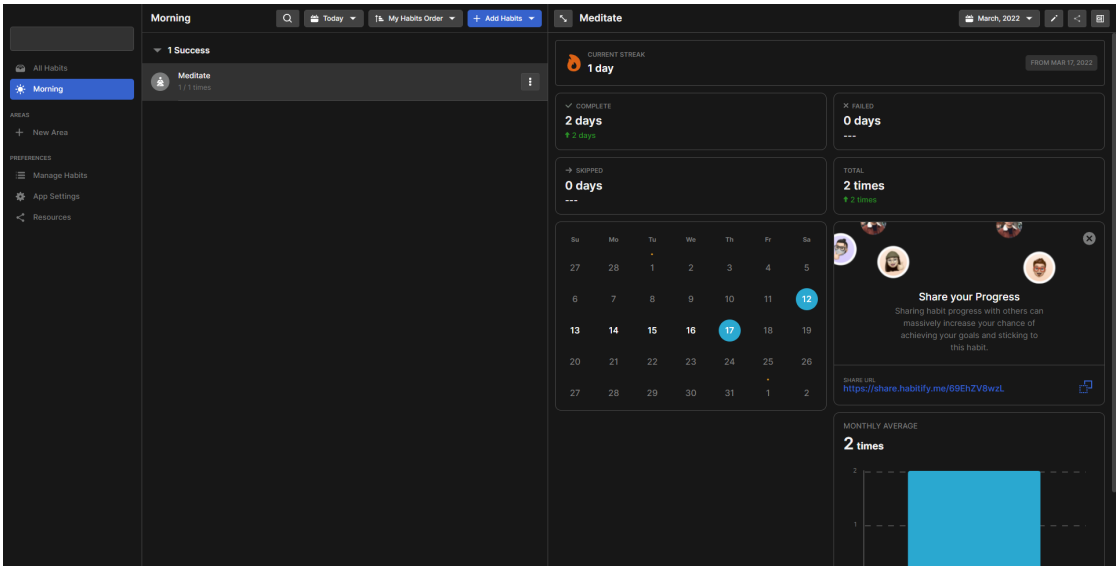
Obrázek 1.3 Habitica [6]

Habitica je bezkonkurenční, co se týče gamifikace. Člověk, který má v oblibě RPG hry si určitě oblíbí tuto aplikaci a pomůže mu s dosažením jeho osobních cílů. Aplikace také poskytuje sociální aspekty, např. různá fóra anebo možnost pozvat kamarády na společné úkoly.

Na druhou stranu, člověk zaměřený na data a statistiky bude zklamán, protože aplikace v této branži neexceluje. Dále je náročné se na začátku zorientovat v uživatelském rozhraní a všech funkcionalitách jako jsou například systém s mincemi, všechny druhy položek v inventáři apod.

1.2.2 Habitify

Habitify poskytuje v základní verzi funkcionality jako je: definice aktivit, kategorizaci aktivit, jednoduché statistiky. Poskytuje také prémiovou verzi, která nabízí neomezené aktivity, systém upozornění, zaznamenávání nálady, poznámky k aktivitám apod. V aplikace je snadné se zorientovat díky jednoduchému a modernímu uživatelskému rozhraní.



Obrázek 1.4 Habitify [7]

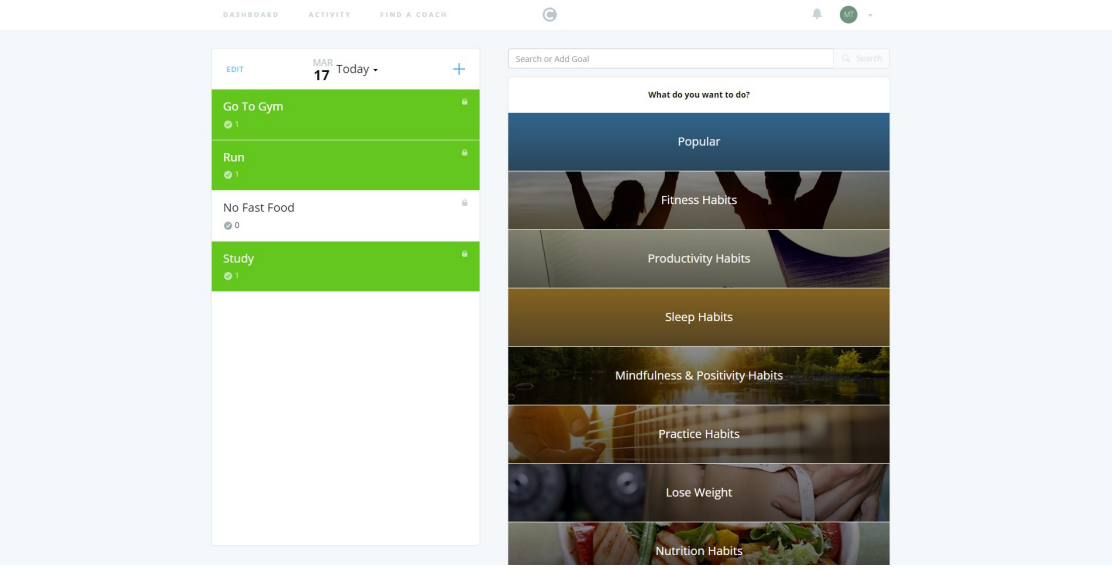
Pro uživatele, který hledá jednoduchou aplikaci, která nabízí základní funkcionality pro sledování aktivit, je Habitify jednou z možností. V aplikaci se uživatel rychle zorientuje a může začít.

Co se týče záporů, aplikace v základní verzi je jednoduchá a neposkytuje jiné zajímavé funkcionality. Tyto funkcionality jsou schované za prémiovou verzí, která stojí reálné peníze.

1.2.3 Coach.me

Aplikace Coach.me poskytuje kromě základních funkcionalit habit trackingu také placené koučování. Uživatel si může zažádat o kouče k předem definovaným aktivitám. Nabízí jednoduché uživatelské rozhraní a výběr z předem definovaných aktivit. Jednotlivé aktivity mají diskuzní fóra k danému tématu aktivity. Uživatelská aktivita je ve výchozím nastavení veřejná, takže ostatní uživatelé můžou vidět, pokud má uživatel splněnou nebo nesplněnou aktivitu.
Na druhou stranu aplikace neposkytuje různé statistiky k aktivitám a uživatel si nemůže definovat vlastní aktivity. Aplikace má také jednoduché uživatelské rozhraní a poskytuje pouze základní funkcionality, co se týče habit trackingu.

Aplikace je primárně zaměřená na placené koučování od ostatních uživatelů. Uživatelé si můžou vytvořit profil kouče a ostatní uživatelé se zapíšou ke koučovi



Obrázek 1.5 Coach.me [8]

a zanechat hodnocení. I přes uživatelské hodnocení koučů, aplikace negarantuje špičkovou službu všech koučů, protože se koučem může stát kdokoliv.

1.2.4 Shrnutí analýzy existujících řešení

Na základě analýzy existujících řešení, lze identifikovat zápory jednotlivých aplikací. Vývoj aplikace se tedy zaměří na vyvarování se těchto záporů jako jsou: příliš jednoduché statistiky, nedostatek prvků gamifikace, nedostatek sledovaných hodnot aktivit a složitost uživatelského rozhraní.

1.3 Analýza funkčních a nefunkčních požadavků

Pro bližší pochopení aplikace je nutné si projít zanalyzované existující řešení a na základě poznatků si sestavit funkční a nefunkční požadavky.

1.3.1 Funkční požadavky

Funkční požadavky definují chování systému za konkrétních podmínek a jaké jsou jeho vlastnosti a funkce(popisují, co má aplikace dělat). Mezi tyto požadavky patří byznys požadavky, administrativní funkce, autentizace uživatelů apod. [9]

F1: Autentizace uživatele

Priorita: Vysoká
Uživatel se může zaregistrovat, přihlásit a odhlásit. Po zaregistrování bude uživateli poslán autentizační odkaz na email zadaný v registračním formuláři. Po úspěšném ověření emailové adresy, se bude moct uživatel přihlásit.

F2: Správa profilu uživatele

Priorita: Vysoká
Přihlášený uživatel bude mít přístup do svého profilu, kde si bude moct změnit emailovou adresu(uživatel musí ověřit novou emailovou adresu), změnit profilovou fotku a změnit heslo.

F3: Vytvoření aktivity

Priorita: Vysoká
Přihlášený uživatel si bude moct definovat svoje aktivity, na které by se chtěl zaměřit. Při vytváření aktivity si může nastavit časový interval a dny, ve kterých by danou aktivitu chtěl dělat.

F4: Správa aktivit

Priorita: Vysoká
Přihlášený uživatel si bude moct změnit a smazat dříve vytvořené aktivity. Bude mít možnost si změnit časové období a dny v týdnu, kdy se aktivita bude opakovat. Po změnění se aktivita přegeneruje podle hodnot, které si uživatel nastavil. Uživatel bude moct vytvořené aktivity označit jako splněné/nesplněné a popřípadě nastavit obtížnost začít danou aktivitu a obtížnost dokončení aktivity.

F5: Přehled aktivit

Priorita: Střední
Přihlášený uživatel bude mít možnost si prohlédnout stav všech vytvořených aktivit v daném týdnu nebo měsíci. Přehled aktivit bude zobrazen pomocí tabulky v „kalendářním“ formátu. Aplikace uživateli umožní procházet mezi jednotlivými týdny a měsíci. Kliknutím na jednotlivé buňky si bude moct uživatel editovat stav aktivity.

F6: Statistiky aktivity

Priorita: Střední
Přihlášený uživatel bude mít možnost si sledovat různé statistiky ke konkrétní aktivitě. Statistiky budou generovány dynamicky na základě různých metrik, které si uživatel zvolí při změně stavů aktivity.

F7: Statistiky všech aktivit

Priorita: Střední
Přihlášený uživatel bude mít možnost si sledovat celkové statistiky všech aktivit. Uživatel se bude moct podívat na statistiky měsíční, roční a celkové. Také mu bude umožněno procházet mezi jednotlivými časovými obdobími. Statistiky budou generovány dynamicky na základě různých metrik, které si uživatel zvolí při změně stavů aktivit.

Mezi typy statistik bude patřit: procentová úspěšnost splnění aktivit podle dnů, procentová úspěšnost splnění podle aktivit, kompozice podle jednotlivých aktivit, průměrná obtížnost začít aktivity a průměrná obtížnost dokončit aktivity.

F8: Systém úrovní uživatele

Priorita: Střední
Za každou aktivitu označenou jako splněnou se zvýší zkušenosti uživatele a po dosažení kapacity současné úrovně se uživateli zvýší jeho úroveň. Při označení aktivity jako nesplněnou se uživateli sníží zkušenosti uživatele a po překročení dolní hranice(zde 0) současné úrovně se uživateli sníží jeho úroveň. Při zvýšení úrovně se uživateli také zvýší kapacita zkušeností další úrovně.

F9: Systém milníků

Priorita: Střední
Po vytvoření aktivity se uživateli vytvoří milníky. Podporované druhy milníků budou počet splnění aktivity a počet splnění aktivity za sebou. Při označení aktivity jako splněné/nesplněné se zaznamená pokrok ke splnění daného milníku aktivity.

F10: Přehled milníků

Priorita: Střední
Přihlášený uživatel bude mít možnost si prohlédnout přehled milníků a jejich pokrok ke splnění daného milníku. Uživateli bude umožněno filtrovat milníky podle jednotlivých aktivit.

F11: Systém notifikací

Priorita: Nízká
Po splnění milníku nebo zvýšení úrovně uživatele bude uživatel upozorněn pomocí systému notifikací.

1.3.2 Nefunkční požadavky

Nefunkční požadavky slouží jako omezení návrhu systému a specifikují kvalitativní atributy systému. [10]

NF1: Responsivita

Aplikace bude podporovat různá rozlišení zařízení jako jsou například: desktop, laptop a mobilní zařízení.

NF2: Webová aplikace

Aplikace bude vyvíjena jako webová aplikace, pro lehkou dostupnost.

NF3: Rozšířitelnost

Aplikace bude vyvíjena tak, aby umožnila rozšířeni funkcionalit v budoucnosti.

NF4: Bezpečnost

Aplikace bude mít zabezpečené vstupy proti klasickým webovým útokům a bude správně kontrolovat vstupní hodnoty.

1.4 Případy užití

Případy užití tvoří základ toho, jak systém interaguje s vnějšími prvky kolem něj: aktéři/uživatelé, jiné systémy a dalšími faktory(datum/čas, zvláštní podmínky prostředí, ...). Případy užití jsou textové popisy interakce mezi některými vnějšími aktéry a systémem. [11]

Následující případy užití vznikly na základě funkčních požadavků. U každého případů užití definuji název, hlavní aktéry, popis a scénář akcí.

UC1: Registrace uživatele

Aktéři: Nepřihlášený uživatel
Popis: Nový uživatel se chce zaregistrovat.
Scénář akcí: Na domovské stránce se bude nacházet registrační formulář uprostřed stránky. Nový uživatel vyplní přihlašovací jméno, emailovou adresu a heslo. Po odeslání formuláře, server ověří vstupní data a pošle na zadaný email autentizační odkaz.

Po kliknutí na autentizační odkaz se ověří emailová adresa a uživatel bude přihlášen. V případě neplatných vstupních dat je uživateli zobrazena chybová hláška.

UC2: Přihlášení uživatele

Aktéři: Nepřihlášený uživatel
Popis: Nepřihlášený uživatel se chce přihlásit.
Scénář akcí: V horním pravém rohu bude tlačítko *Login* pro přihlášení. Poté se zobrazí přihlašovací formulář. Uživatel vyplní přihlašovací jméno a heslo.

Po odeslání formuláře, server ověří vstupní data a zkontroluje jestli má ověřený email. V případě neplatných vstupních dat je uživateli zobrazena chybová hláška.

UC3: Obnovení hesla

Aktéři: Nepřihlášený uživatel
Popis: Nepřihlášený uživatel si chce obnovit heslo.
Scénář akcí: V horním pravém rohu bude tlačítko *Login* pro přihlášení. Poté se zobrazí přihlašovací formulář, na kterém bude odkaz *Forgotten password* pro obnovení hesla. Uživatel vyplní emailovou adresu.

Po odeslání formuláře, server ověří vstupní data a pošle na zadaný email odkaz pro obnovení hesla. Po kliknutí na odkaz si bude moct uživatel nastavit nové heslo.

UC4: Editace profilu uživatele

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce změnit heslo nebo profilový obrázek.
Scénář akcí: Po přihlášení si může uživatel otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Profile* bude uživatel přesměrován do profilové sekce.

1. Pokud si chce uživatel změnit heslo, zvolí možnost *Change password*. Uživatel vyplní současné heslo a potvrdí nové heslo. Po odeslání formuláře, server ověří vstupní data. V případě neplatných vstupních dat je uživateli zobrazena chybová hláška.

2. Pokud si chce uživatel změnit profilový obrázek, klikne na tlačítko *Choose file* pro nahrání souboru pod profilovým obrázkem. Následně se otevře okénko pro volbu souboru k nahrání jako nový profilový obrázek.

UC5: Změna emailu uživatele

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce změnit současnou emailovou adresu.
Scénář akcí: Po přihlášení si může uživatel otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Profile* bude uživatel přesměrován do profilové sekce.

Pokud si chce uživatel změnit emailovou adresu, zvolí možnost *Change email*. Uživatel vyplní novou emailovou adresu.

Po odeslání formuláře, server ověří vstupní data a pošle na zadaný email autentizační odkaz.

Po kliknutí na autentizační odkaz se ověří emailová adresa a uživateli se nastaví nová adresa. V případě neplatných vstupních dat je uživateli zobrazena chybová hláška.

UC6: Vytvoření aktivity

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce vytvořit novou aktivitu.
Scénář akcí: Po přihlášení si může uživatel otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Activities* bude uživatel přesměrován do výpisu vytvořených aktivit.

Pro vytvoření nové aktivity, uživatel klikne na tlačítko *+ New*, které je napravo od nadpisu *Activities*. Po kliknutí se otevře formulář pro vytvoření nové aktivity. Uživatel zadá startovní datum a koncové datum intervalu, kdy se mají aktivity generovat, zvolí dny v týdnu a potvrdí výběr tlačítkem *Submit*.

Po vytvoření aktivity se uživateli vytvoří milníky. V případě neplatných vstupních dat je uživateli zobrazena chybová hláška.

UC7: Editace aktivity

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce editovat existující aktivitu.
Scénář akcí: Po přihlášení si může uživatel otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Activities* bude uživatel přesměrován do výpisu vytvořených aktivit.

Uživatel si v tabulce výpisu aktivit vybere řádek s aktivitou, kterou chce editovat a klikne na tlačítko s ikonkou pro editaci v posledním sloupci. Po kliknutí se otevře formulář pro editaci zvolené aktivity. Uživatel může editovat startovní datum a koncové datum intervalu, kdy se mají aktivity vygenerovat, zvolí dny v týdnu, vybere si jestli chce prodloužit interval, zkrátit interval nebo přegenerovat existující aktivity a potvrdí výběr tlačítkem *Submit*.

V případě neplatných vstupních dat je uživateli zobrazena chybová hláška.

UC8: Zobrazit přehled aktivit

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce prohlédnout stav aktivit v týdnu/měsíci.
Scénář akcí: Otevřením navigační postranní lišty pomocí tlačítka v horním levém rohu a následně kliknutím na možnost *Home* bude uživatel přesměrován do tabulky s přehledem všech aktivit v daném týdnu. Kliknutím na šipky nad tabulkou může uživatel navigovat mezi jednotlivými týdny.

Kliknutím na tlačítko *Monthly* bude uživatel přesměrován do tabulky s přehledem všech aktivit v daném měsíci.

UC9: Změna stavu aktivity

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce změnit stav aktivity.
Scénář akcí: Po přihlášení si může uživatel změnit stav aktivity na těchto místech:

1. Otevřením postranní lišty pro navigaci pomocí tlačítka v horním levém rohu a následně kliknutím na možnost *Home* bude uživatel přesměrován do tabulky s přehledem všech aktivit v daném týdnu. Kliknutím na buňku v tabulce se otevře vyskakovací okno pro editaci aktivity v daný den. Uživatel si zvolí obtížnost začít danou aktivitu, obtížnost dané aktivity, stav splněná nebo nesplněná aktivita a potvrdí svojí volbu tlačítkem *Submit*.

2. Otevřením postranní lišty pro navigaci pomocí tlačítka v horním levém rohu a následně kliknutím na možnost *Home*. Kliknutím na tlačítko *Monthly* bude uživatel přesměrován do tabulky s přehledem všech aktivit v daném měsíci. Kliknutím na buňku v tabulce se otevře vyskakovací okno pro editaci aktivity v daný den. Uživatel si zvolí obtížnost začít danou aktivitu, obtížnost dané

aktivity, stav splněná nebo nesplněná aktivita a potvrdí svojí volbu tlačítkem *Submit*.

3. Uživateli se po přihlášení zobrazí horní lišta s aktivitami pro daný den. Uživatel si zvolí obtížnost začít danou aktivitu, obtížnost dané aktivity, stav splněná nebo nesplněná aktivita a potvrdí svojí volbu tlačítkem *Submit*.

UC10: Zobrazit statistiky aktivity

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce prohlédnout statistiky existující aktivity.
Scénář akcí: Po přihlášení si může uživatel otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Activities* bude uživatel přesměrován do výpisu vytvořených aktivit.

Uživatel si v tabulce výpisu aktivit vybere řádek s aktivitou, u které chce vidět statistiky a klikne na tlačítko s ikonkou pro zobrazení statistik v posledním sloupci.

UC11: Zobrazit celkové statistiky

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce prohlédnout statistiky všech existujících aktivit.
Scénář akcí: Po přihlášení si uživatel může otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Statistics* bude uživatel přesměrován do celkových statistik všech existujících aktivit.

1. Pokud se chce uživatel podívat na měsíční statistiky, klikne na tlačítko *Monthly* ve výběru typu statistik nad vygenerovanými grafy.
2. Pokud se chce uživatel podívat na roční statistiky, klikne na tlačítko *Yearly* ve výběru typu statistik nad vygenerovanými grafy.
3. Pokud se chce uživatel podívat na celkové statistiky, klikne na tlačítko *All time* ve výběru typu statistik nad vygenerovanými grafy.

Pokud se chce uživatel podívat na předchozí nebo následující měsíc/rok, lze navigovat pomocí šípek nad vygenerovanými grafy.

UC12: Vytvoření milníků

Aktéři: Přihlášený uživatel
Popis: Server vytvoří milníky poté, co uživatel vytvoří novou aktivitu.
Scénář akcí: Přihlášený uživatel dokončí případ užití UC6 a server vytvoří pro uživatele milníky typu *Počet splnění aktivity* a *Počet splnění aktivity za sebou*.

UC13: Aktualizace milníků

Aktéři: Přihlášený uživatel
Popis: Server aktualizuje milníky poté, co uživatel změní stav aktivity.

Scénář akcí: Přihlášený uživatel dokončí případ užití UC9 a server aktualizuje pokrok ke splnění milníků aktualizované aktivity na základě změny stavu aktivity:

1. Splněný → nesplněný:
   - Pokrok ke splnění milníků typu *Počet splnění aktivity* se sníží o 1.
   - Pokrok ke splnění milníků typu *Počet splnění aktivity za sebou* se znovu přepočítá(server najde novou nejdelší sérii splněných aktivit za sebou).
2. Nesplněný → splněný:
   - Pokrok ke splnění milníků typu *Počet splnění aktivity* se zvýší o 1.
   - Pokrok ke splnění milníků typu *Počet splnění aktivity za sebou* se znovu přepočítá(server najde novou nejdelší sérii splněných aktivit za sebou).

Při splnění milníku se milník označí jako splněný.

UC14: Zobrazit přehled milníků

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce prohlédnout přehled milníků.
Scénář akcí: Po přihlášení si může uživatel otevřít navigační postranní lištu pomocí tlačítka v horním levém rohu. Po kliknutí na možnost *Achievements* bude uživatel přesměrován do přehledu milníků.

Uživatel si může filtrovat milníky podle jednotlivých aktivit.

UC15: Aktualizace úrovně uživatele

Aktéři: Přihlášený uživatel
Popis: Server aktualizuje zkušenost uživatele(popřípadě úroveň uživatele) poté, co uživatel změní stav aktivity.
Scénář akcí: Přihlášený uživatel dokončí případ užití UC9 a server aktualizuje zkušenost uživatele(popřípadě úroveň uživatele) na základě změny stavu aktivity:

1. Splněný → nesplněný: zkušenost uživatele se sníží o 1. Pokud zkušenost uživatele dosáhne -1 současné úrovně, uživateli se sníží úroveň o 1.
2. Nesplněný → splněný: zkušenost uživatele se zvýší o 1. Pokud zkušenost uživatele dosáhne kapacity současné úrovně, uživateli se zvýší úroveň o 1.

UC16: Vytvoření notifikací

Aktéři: Přihlášený uživatel
Popis: Server vytvoří notifikace poté, co uživatel splní milník nebo dosáhne nové úrovně.
Scénář akcí: Server vytvoří notifikace pro uživatele pokud:

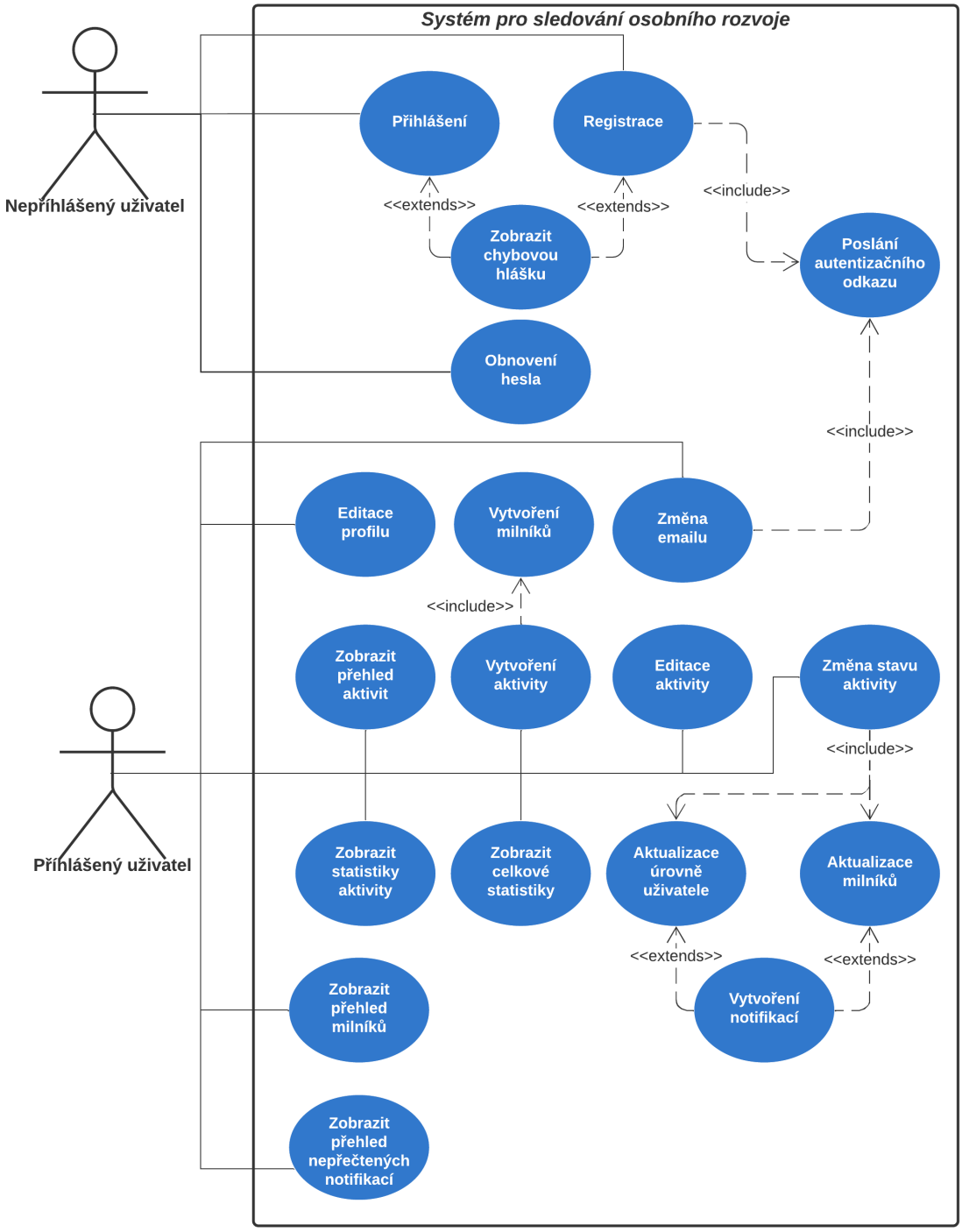
1. Přihlášený uživatel dokončí případ užití UC13 a označí milník jako splněný.
2. Přihlášený uživatel dokončí případ užití UC15 a uživateli se zvýší úroveň.

UC17: Zobrazit přehled nepřečtených notifikací

Aktéři: Přihlášený uživatel
Popis: Přihlášený uživatel si chce přečíst upozornění.
Scénář akcí: Po přihlášení si může uživatel prohlédnout upozornění v horním levém rohu, kliknutím na ikonku se zvonkem.

Tabulka 1.1 Pokrytí funkčních požadavků

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
UC1	X										
UC2	X										
UC3	X										
UC4		X									
UC5		X									
UC6			X								
UC7				X							
UC8					X						
UC9				X							
UC10						X					
UC11							X				
UC12									X		
UC13									X		
UC14										X	
UC15								X			
UC16											X
UC17											X



Obrázek 1.6 Diagram případů užití

Kapitola 2
Návrh

V softwarovém inženýrství je fáze návrhu důležitá zejména kvůli výběru vhodných technologií(platforma, back end, front end apod.) pro splnění požadavků vzešlých z fáze analýzy.
V této kapitole nejprve představím návrhové vzory MVC a MVT. Následně popíšu vybranou back-end technologii Django(web framework psaný v programovacím jazyce Python), databáze podporované Django frameworkem, databázový model aplikace a použité front-end technologie.

2.1 Návrhový vzor MVC

Model-View-Controller(MVC) je návrhový vzor, který rozděluje logiku aplikace do 3 hlavních komponent(Model, View a Controller). Tento vzor pomáhá rozdělit odpovědnost mezi jednotlivé komponenty.

Použitím vzoru MVC jsou požadavky uživatele směřovány komponentě Controller, která spolupracuje s komponentou Model. Hlavním úkolem komponent Model je zpracování požadavků uživatele a případné výsledky dotazů do databáze. Controller následně vybere vhodný View, předá mu data zpracované komponentou Model a nakonec se uživateli zobrazí daná View komponenta se zpracovanými daty. [12]

Model

V této komponentě probíhá byznys logika a s ní spojené operace jako například: výpočty, databázové dotazy, validace apod.

View

View má na starost prezentaci obsahu(data předána komponentou Model) přes uživatelské rozhraní. V této komponentě může probíhat logika spojená s prezentací obsahu.

Controller

Controller jsou komponenty, které mají na starost požadavky uživatelů. Na základě těchto požadavků vybírají vhodné komponenty Model a následně vyberou vhodnou komponentu View k prezentaci.

2.2 Návrhový vzor MVT

Dle článku [13] je Model-View-Template(MVT) návrhový vzor používaný frameworkem Django a je odvozen od návrhového vzoru MVC.

Model

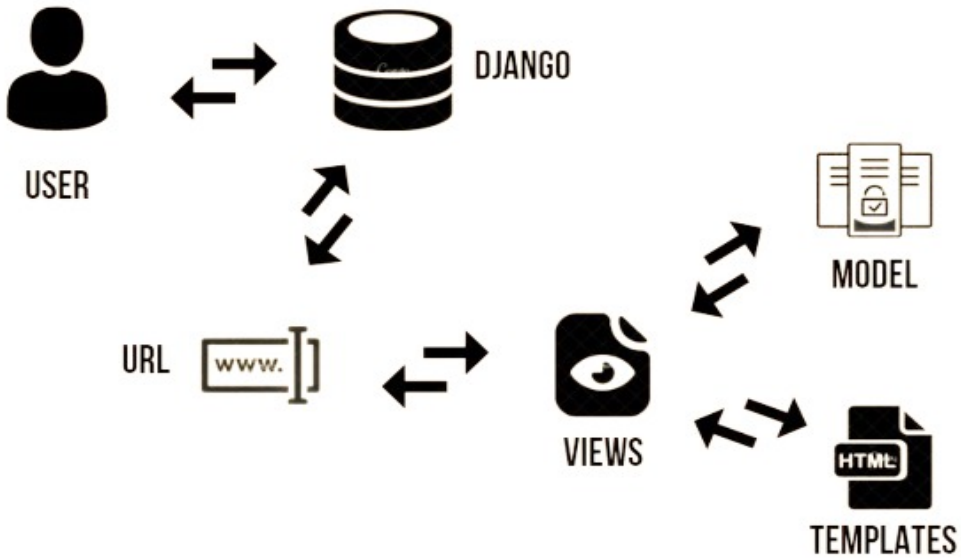
Model komponenty poskytují rozhraní pro uložená data v databázi. Jsou zodpovědné za údržbu dat a jejich manipulaci.

View

View komponenty určují data, která jsou prezentována uživateli. Nejedná se o to jak data vypadají, ale která data jsou prezentována. Tato komponenta předá data z Model komponent příslušným Template komponentám, které je zobrazí uživateli.

Template

Template komponenty jsou plně zodpovědné za prezentaci obsahu přes uživatelské rozhraní. V této komponentě může probíhat logika spojená s prezentací obsahu.



Obrázek 2.1 Diagram návrhového vzoru MVT, používané frameworkem Django. [13]

2.3 Použité back-end technologie

2.3.1 Django

Django je volně přístupný open-source webový framework napsaný v programovacím jazyce Python. Framework je založen na návrhovém vzoru MVT. Jeho cílem je pomoct vývojářům rychle převést koncepty aplikací do finálního produktu tím, že poskytuje rychlé a snadné řešení pro běžné úlohy vývoje webu.

Mezi poskytované funkcionality patří: [14]

- **Object-relational mapper:**
  Django umožňuje definovat vlastní datové modely. Každý atribut modelu představuje databázové pole. Díky tomu Django poskytuje automaticky generované API pro přístup k databázi. V případě potřeby lze stále psát vlastní SQL dotazy.
- **URL mapování na View komponenty:**
  Django umožňuje mapovat url adresy přes Python modul nazvaný URLconf, ve kterém si lze definovat jednoduché mapování mezi vzory URL adres a View komponentami a může odkazovat na jiná mapování.
- **Template komponenty:**
  Django poskytuje značkovací jazyk Django template language (DTL), který poskytuje pohodlný způsob generování dynamického HTML. Nejběžnější přístup je založen na šablonách, které obsahují statické části(HTML, CSS) a také některé speciální syntaxe popisující, jak bude vkládán a zpracován dynamický obsah(většinou se jedná o data předaná z View komponent). Další funkcionalitou je také flexibilita a rozšiřitelnost značkovacího jazyka, což umožňuje vývojářům rozšířit jazyk šablon podle potřeby.
- **Formuláře:**
  Django poskytuje výkonnou knihovnu pro práci s formuláři, která zpracovává vykreslování formulářů jako HTML, ověřuje data odeslaná uživateli a převádí tato data na nativní typy v jazyce Python. Django také poskytuje způsob, jak generovat formuláře z existujících Modelů a používat tyto formuláře k vytváření a aktualizaci dat daného Modelu.
- **Autentizace uživatelů:**
  Django přichází se systémem pro autentizaci uživatelů. Zpracovává uživatelské účty, skupiny, oprávnění, uživatelské relace založené na souborech cookie, konfigurovatelný systém hašování hesel. Pro dodatečné autentizační funkcionality lze použít balíčky z třetích stran.
- **Admin panel:**
  Jednou z nejzajímavějších částí frameworku Django je jeho administrátorské rozhraní, které se vytvoří automaticky. Čte metadata existujících Modelů a poskytuje výkonné a produkční rozhraní, které mohou správci okamžitě použít k zahájení správy obsahu na vyvíjeném webu.

- **Internacionalizace:**
  Django nabízí plnou podporu pro překlad textu do různých jazyků a formátování dat, časů, čísel a časových pásem specifické pro různé země. Umožňuje vývojářům určit, které části jejich aplikací by měly být přeloženy nebo formátovány pro místní jazyky a kultury. Tyto funkcionality lze použít k lokalizaci webových aplikací pro konkrétní uživatele podle jejich preferencí.
- **Bezpečnost:**
  Django pomáhá vývojářům vyhnout se mnoha běžným bezpečnostním chybám, jako jsou SQL injection, cross-site scripting, cross-site request forgering a clickjacking. Také poskytuje systém ověřování uživatelů a bezpečný způsob správy uživatelských účtů a hesel.

Při výběru back-end technologie jsem se rozhodoval mezi Django, Spring Boot a PHP. Django jsem si vybral z důvodu jednoduchosti, rychlosti vývoje webových aplikací a hlavně kvůli předem zmíněných funkcionalit, které framework poskytuje. Dle mého názoru má Django nejjednodušší workflow z těchto 3 technologií díky přehledné struktuře adresáře, která se generuje automaticky a každý soubor jednoznačně odděluje logické celky.

2.3.2 Databáze

Ve výchozí instalaci Django frameworku je použita SQLite databáze, která umožňuje ihned pracovat s databází po instalaci frameworku. Uživatel si může změnit databázi v konfiguračním souboru.

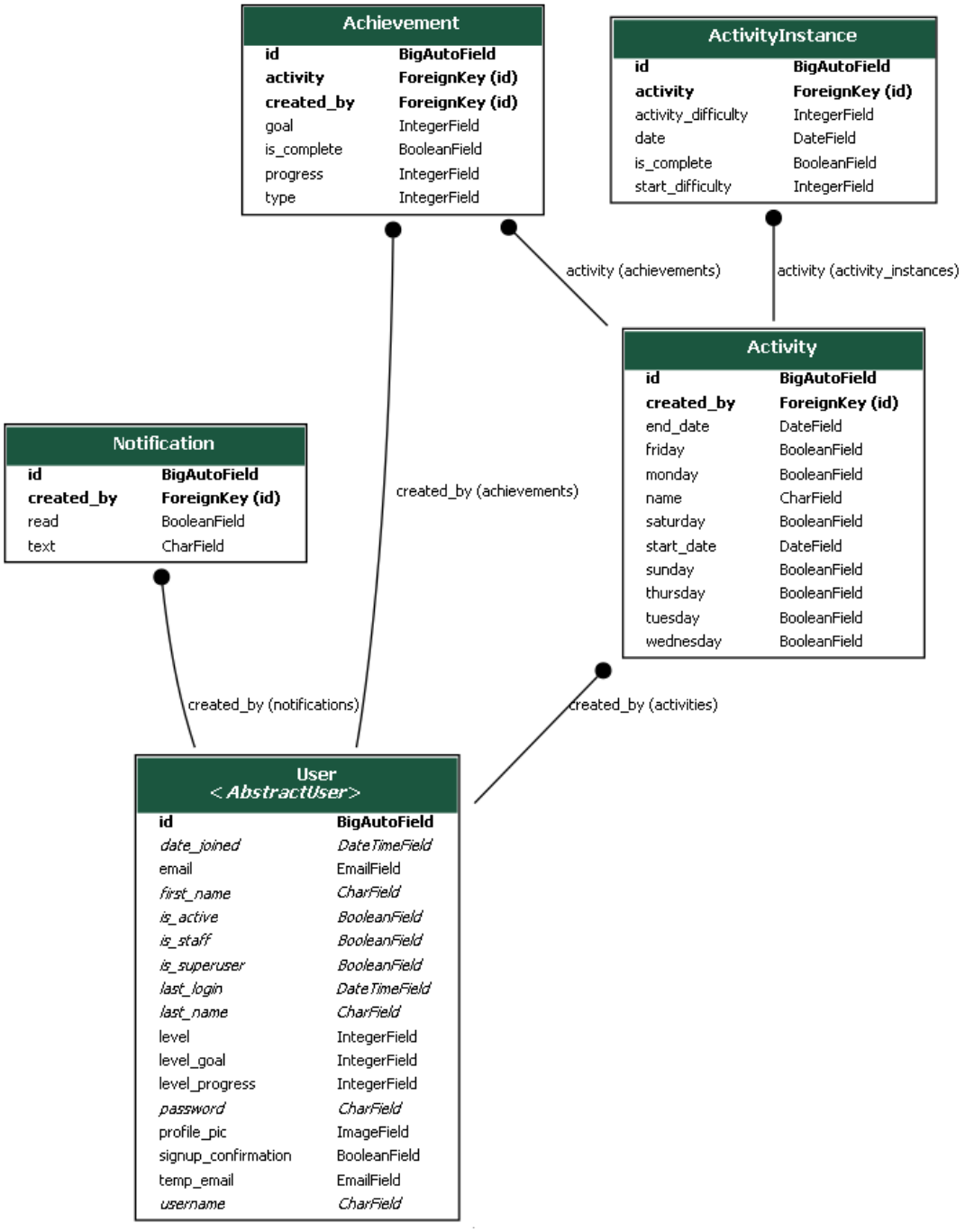
Podporované databáze frameworkem Django: [15]

- PostgreSQL
- MariaDB
- MySQL
- Oracle
- SQLite
- Existuje také řada databázových back-end řešení poskytovaných třetími stranami.

Pro svojí implementaci jsem se rozhodl pracovat s databází PostgreSQL, protože se často používá v produkci Django webových aplikací, a proto existuje mnoho zdrojů pro konfiguraci a správu této databáze. Dalším důvodem je také bezpečnost. U PostgreSQL lze nastavit různá práva uživatelským účtům na rozdíl od SQLite, kde si nelze nastavit práva v databázové vrstvě, ale pouze v aplikační vrstvě. PostgreSQL je pokročilá verze SQL, která poskytuje podporu pro různé funkce SQL jako například: cizí klíče, komplexní dotazy, triggers, aktualizovatelné pohledy, integrita transakcí apod. [16]

2.3.3 Databázový model

Databázový model je vygenerovaný z existujících modelů v aplikaci pomocí balíčku django-extensions. Je nutné si stáhnou vykreslovací knihovnu Graphviz/Pydot. Pro instalaci na Linux lze pokračovat podle [17]. Pro instalaci na Windows je nutné si stáhnout instalační program z oficiálních stránek Graphviz. [18]



Obrázek 2.2 Databázový model

Databázový diagram se liší od doménového modelu 1.2 tím, že rozlišuje aktivitu(Activity) a instanci aktivity(ActivityInstance). To je z důvodu udržení historie zvolených hodnot pro vygenerování instancí dané aktivity.

Dalším rozšířením je entita Notification, která slouží pro splnění funkčního požadavku systému notifikací(viz. 1.3.1).

Pro splnění požadavku ze zadaní ohledně možnosti sledovat alespoň 3 hodnot u aktivit jsem přidal do entity ActivityInstance následující atributy:

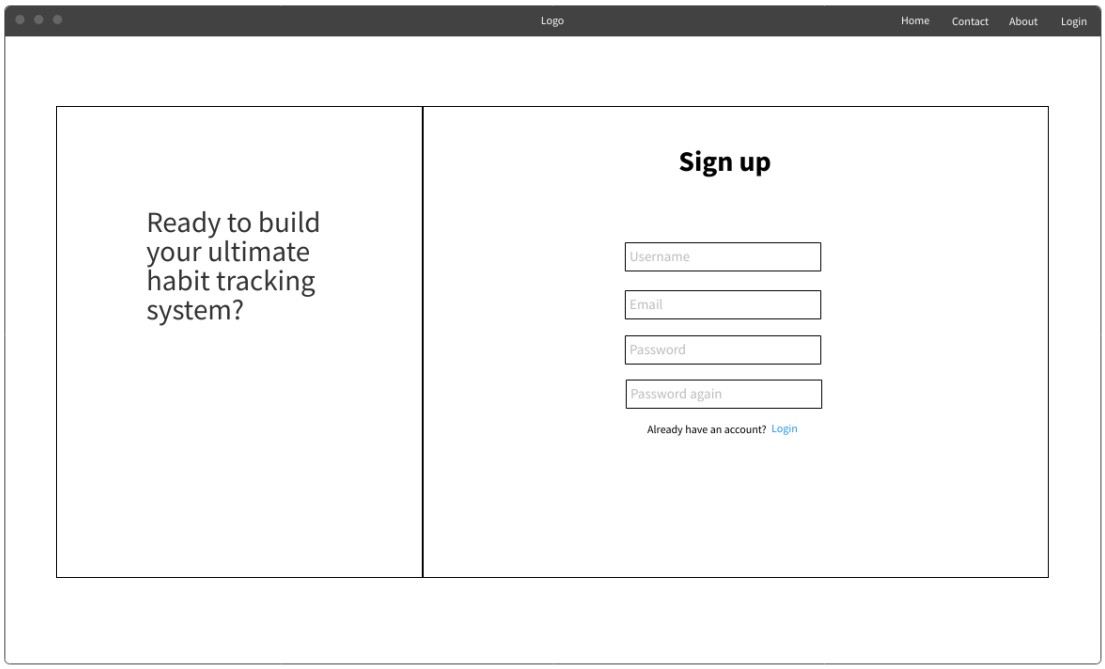
- **is_complete :** slouží pro sledování splnění aktivity
- **start_difficulty :** slouží pro sledování obtížnosti začít danou aktivitu
- **activity_difficulty :** slouží pro sledování obtížnosti dané aktivity

2.4 Návrh UI

První dojem je nedílnou součástí lidské psychologie, a proto je návrh uživatelského rozhraní technologická disciplína sama o sobě. V dnešní době si mnoho uživatelů vybírá aplikace na základě vzhledu uživatelského rozhraní a uživatelské přívětivosti. Z těchto důvodů jsem se rozhodl použít populární designové vzory, na které jsou uživatelé zvyklí.

2.4.1 Domovská stránka

Domovská stránka je to první, co uživatelé uvidí. Slouží jako centrální navigační místo, odkud se uživatelé mohou dostat do jiných oblastí aplikace.



Obrázek 2.3 Domovská stránka - wireframe

Při návrhu jsem se rozhodl, že na domovské stránce přidám motivační text a registraci uživatele.

Pro vyřešení responsivity na malých zařízení jsem se rozhodl, že ponechám horní navigační lištu, pouze změním obsah domovské stránky na zobrazení pouze registračního/přihlašovacího formuláře. Tímto se zaručí, že zmenšením šířky obrazovky nevzniknou webové komponenty, které nejdou přečíst kvůli velikosti písma.

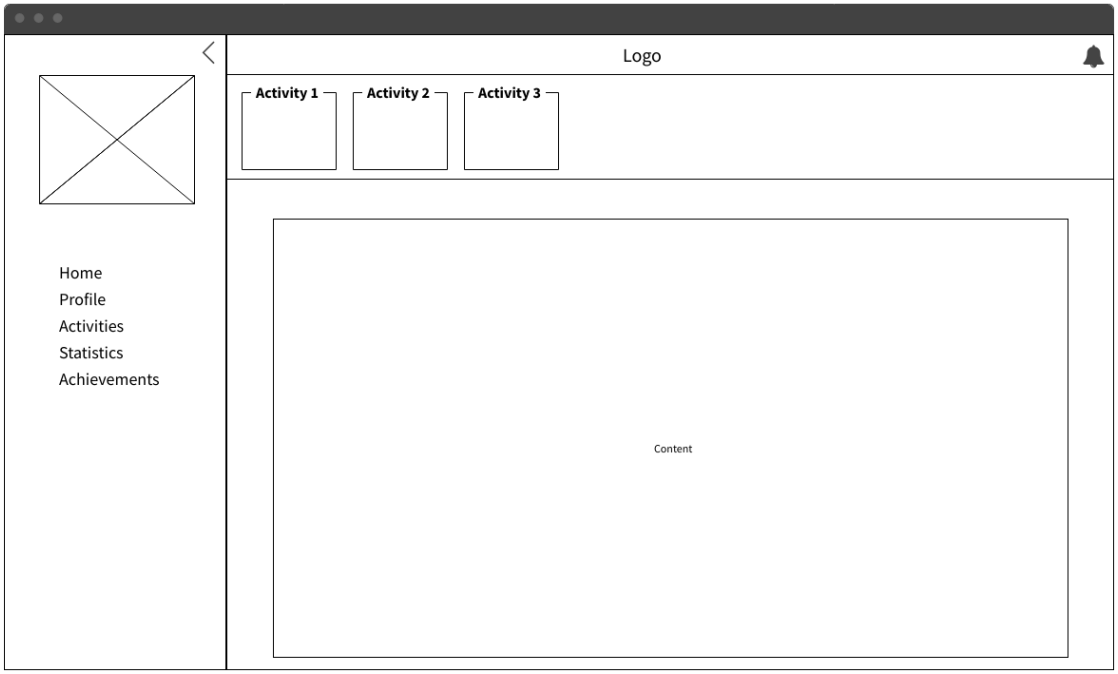
2.4.2 Rozhraní přihlášeného uživatele

Rozhraní pro přihlášené uživatele je jádro aplikace. Zde uživatelé stráví nejvíce času používáním funkcionalit aplikace, proto je této části věnována největší pozornost.

Při návrhu jsem se rozhodl pro jednoduchý a uživatelsky přívětivý design. Pro navigaci mezi funkcionalitami jsem se rozhodl použít otevírací postranní navigační lištu. Součástí této lišty bude i profilový obrázek uživatele, který si bude moct sám nastavit.

V horní části bude lišta, která bude obsahovat logo aplikace a v pravém rohu ikonku pro zobrazení upozornění. Pod touto lištou se bude nacházet lišta s aktivitami pro daný den.

Obsah(content) stránky bude vedle postranní navigační lišty a pod horními lištami. Každá stránka si bude moct navrhnout obsah(bude dynamický).



Obrázek 2.4 Rozhraní přihlášeného uživatele - wireframe

Pro vyřešení responsivity na malých zařízení jsem se rozhodl, že otevřený postranní bar zabere celou obrazovku místo toho, aby otevřením/zavřením postranní lišty se obsah stránky zmenšil/zvětšil. Tímto postranní navigační lišta nebude překážet na zařízeních s malých rozlišením.

Responsivitu jednotlivých komponent v obsahu jsem se rozhodl vyřešit pomocí grid systému, který na základě šířky obrazovky rozhodne kolik komponent se celkově vejde na jeden řádek. Pokud se nějaká komponenta nevejde na řádek, vyrenderuje se na dalším řádku.

2.5 Použité front-end technologie

V dnešní době se zřídka kdy potkáte s webovou aplikací, která by nepoužívala HTML, CSS nebo JavaScript. Jedná se o populární technologie, které jsou aktivně aktualizovány a udržují si silnou komunitu programátorů. Z těchto důvodů jsem pro vývoj vzhledu klientské části aplikace vybral tyto technologie.

2.5.1 HTML

HyperText Markup Language(HTML) je stavebním kamenem webu. Definuje význam a strukturu webového obsahu. Kromě HTML se ještě používá CSS k definici vzhledu/prezentace webové stránky a JavaScript k implementaci funkčnosti a chování webu.

Hypertext označuje odkazy, které vzájemně propojují webové stránky, a to buď v rámci jednoho webu nebo mezi jinými weby. Odkazy jsou základním aspektem webu.

Markup jsou značky k anotaci textu, obrázků a dalšího obsahu pro zobrazení ve webovém prohlížeči. Mezi některé značky HTML patří: head, title, body, header, footer, article, section, p, div, span, img, ul, ol, li a mnoho dalších.

Element HTML je od ostatního textu v HTML dokumentu oddělen „tagy", které se skládají z názvu značky obklopeného symboly „<"a „>". Mezi některé příklady tagů patří: `<head>`, `<title>`, `<body>`, `<header>` apod. [19]

2.5.2 CSS

Cascading style sheets(CSS) je jedním ze stavebních kamenů webu. Slouží pro definici prezentace/vzhledu webových stránek, barev, rozložení a písem.

Umožňuje přizpůsobit prezentaci obsahu různým typům zařízení jako jsou velké obrazovky, malé obrazovky nebo tiskárny.

CSS je nezávislé na HTML a lze jej použít s jakýmkoliv značkovacím jazykem založeným na XML. Oddělení HTML od CSS usnadňuje údržbu webů, sdílení stylů napříč stránkami a přizpůsobení stránek různým prostředím. Toto je označováno jako oddělení struktury/obsahu od prezentace. [20]

Během návrhu jsem se soustředil na znovupoužitelnost CSS souborů. Snažil jsem se držet CSS a HTML kód oddělený, abych mohl podle potřeby použít stejný CSS soubor u různých stránek.

2.5.3 JavaScript

Dle MDN [21] je JavaScript(JS) interpretovaný, objektově orientovaný programovací jazyk a je známý jako skriptovací jazyk pro webové stránky, ale používá se i v prostředích bez webového prohlížeče. Jedná se o skriptovací jazyk s více paradigmaty založený na prototypech, který je dynamický a podporuje objektově orientované, imperativní a funkční styly programování.

JavaScript běží na klientské straně webu, díky tomu lze navrhnout/naprogramovat, jak se webové stránky chovají při výskytu události. JavaScript je snadno naučitelný a také výkonný skriptovací jazyk, široce používaný pro ovládání chování webových stránek.

2.5.4 AJAX

AJAX je zkratka pro Asynchronous JavaScript And XML. Stručně řečeno, je to použití objektu XMLHttpRequest pro komunikaci se servery. Díky AJAX lze odesílat a přijímat informace v různých formátech, včetně JSON, XML, HTML a textových souborů. Nejatraktivnější charakteristikou AJAX je jeho „asynchronní“ povaha, což znamená, že může komunikovat se serverem, vyměňovat si data a aktualizovat stránku, aniž by bylo nutné stránku obnovovat. [22]

Mezi hlavní funkce AJAX patří:

1. Odeslání požadavků na server bez opětovného načtení stránky.
2. Přijímání dat ze serveru a pracovat s nimi.

2.5.5 jQuery

jQuery je rychlá, kompaktní a na funkce bohatá JavaScript knihovna. Díky snadno použitelnému rozhraní API, které funguje ve velkém množství prohlížečů, je manipulace s dokumenty HTML, zpracování událostí, animace a AJAX, mnohem jednodušší. Díky kombinaci všestrannosti a rozšiřitelnosti změnil jQuery způsob, jakým miliony lidí píší JavaScript. [23]

Knihovnu jQuery lze využít u animace otevírání/zavírání postranní lišty u rozhraní pro přihlášené uživatele. Z nabízených jQuery funkcí lze také využít jQuery implementaci AJAX, která se snadno používá.

2.5.6 Graph.js

Podle oficiálních stránek [24] je Graph.js open-source JavaScript knihovna, která poskytuje responsivní zpracování a vykreslování grafů.

Mezi podporované typy grafů patří:

- plošné grafy,
- sloupcové grafy,

- bublinové grafy,
- výsečové grafy,
- čárové grafy,
- kombinované grafy,
- a další.

Knihovnu lze použít ke zpracování a vykreslování celkových statistik aktivit a statistiky jednotlivých aktivit. Důvodem proč jsem si vybral tuto knihovnu je možnost se podívat do zdrojového kódu knihovny a případně si přizpůsobit nějaké funkcionality. Další výhodou je také zakomponovaná responsivita grafů.

2.5.7 Bootstrap

Bootstrap je výkonná sada nástrojů HTML, CSS a JavaScript pro vytváření webových stránek a aplikací. Je to bezplatný open-source projekt, který se snaží usnadnit implementaci responsivity webových aplikací. Zachovává širokou kompatibilitu s různými prohlížeči a nabízí konzistentní design pomocí znovupoužitelnosti nabízených komponent. [25]

Předem zmíněné body jsou hlavním důvodem, proč jsem si knihovnu Bootstrap vybral. Poskytuje responsivní grid systém, který na základě šířky obrazovky rozhodne, kolik komponent se celkově vejde na jeden řádek. Pomocí této techniky lze lehce implementovat rozložení prvků obsahu stránky podle šířky obrazovky. Z nabízených komponent lze použít modální okna pro konfirmační zprávy uživatelům u rizikových funkcionalit. Dále lze použít dropdown okna, které pomůže při implementaci responsivního okna s notifikacemi.

2.5.8 Font Awesome

Font Awesome je knihovna ikon a sada nástrojů. Obsahuje mnoho ikon, které jsou volně dostupné, avšak placená verze obsahuje mnohonásobně více ikon. Font Awesome poskytuje různé styly ikon. U ikon lze také měnit barvy, animace, rotovat a jiné možnosti modifikace existujících ikon. [26]

Ikon v aplikací si mnoho uživatelů nevšímá, ale když v aplikaci chybí, tak aplikace je vnímána v negativním světle. Z tohoto důvodu jsem se rozhodl přidat do designu ikony, které by posílily vzhled aplikace. K dispozici je mnoho knihoven s ikonami, které v zásadě poskytují stejnou funkcionalitu. Font Awesome jsem si vybral, protože se syntaxí mám zkušenosti a jejich volně dostupné ikony jsou dostačující pro tvorbu webových aplikací.

Kapitola 3
Implementace

Fáze implementace je konvergence poznatků vzešlých z analýzy a vybraných technologií během návrhu. V této fázi lze zjistit, jak kvalitně se provedla analýza a návrh aplikace.
V této kapitole představím implementační překážky a jakým způsobem jsem se rozhodl je vyřešit. U všech problémů přidám ukázku zdrojového kódu pro přidání kontextu.

3.1 Generování aktivit

Uživatel si nastaví dny v týdnu, počáteční a koncové datum časového úseku, ve kterém by chtěl vykonávat danou aktivitu. Nové aktivity se pak vygenerují podle nastavených hodnot.

Vybral jsem si tuto variantu algoritmu, kvůli generování milníků. Pokud předem neznám počet aktivit, tak nedokážu určit počet splnění aktivity, aby se daný milník mohl označit jako splněný.

View komponenty předávají boolovské hodnoty z formuláře pro vytvoření/editaci aktivity funkci `generate_activity` a ta na základě těchto hodnot volá funkci `generate_weekdays` pro daný den v týdnu.

Výpis kódu 3.1 Generování aktivit

```
1  def generate_activity(start_date, end_date, activity, *days):
2      new_activities = []
3      if len(days) != 7:
4          return new_activities
5
6      target_weekday = 1
7      for day in days:
8          if day is True:
9              new_activities += generate_activity_weekday(start_date, end_date,
10                                                         activity, target_weekday)
11         target_weekday += 1
12
13     return new_activities
```

Funkce `generate_weekdays` spočítá počet dní do dalšího dne v týdnu od startovního data a poté vypočítá všechny další dny v týdnu, které spadají do datového intervalu.

Výpis kódu 3.2 Generování aktivity pro den v týdnu

```
def generate_weekday(start_date, end_date, activity, target_weekday):
    date_offset = target_weekday - start_date.isoweekday()
    if date_offset < 0:
        date = start_date + timedelta(7 + date_offset)
    else:
        date = start_date + timedelta(date_offset)
    activities = []
    while date <= end_date:
        new_activity = ActivityInstance(activity=activity, date=date)
        activities.append(new_activity)
        date = date + timedelta(7)

    return activities
```

3.2 Generování milníků

Milníky jsou vytvořeny poté, co se vytvoří/změní aktivita.

Podporované druhy milníků:

1. počet splnění aktivity
2. počet splnění aktivity za sebou

Milníky typu 1 se generují na základě parametru `num_of_equal_parts`, který určuje na kolik stejně velkých částí se má celkový počet aktivit rozdělit pro vytvoření milníků.

Výpis kódu 3.3 Generování milníků pro počet splnění aktivity

```
def create_completion_achievement(new_activity, user, num_of_equal_parts,
                                  total_instances):
    new_achievements = []
    for i in range(1, num_of_equal_parts + 1):
        goal = round((total_instances/num_of_equal_parts)*i)
        new_achievement = Achievement(activity=activity, created_by=user,
                                      type=NUM_OF_COMPLETION, progress=0,
                                      goal=goal)

        new_achievements.append(new_achievement)

    return new_achievements
```

Milníky typu 2 se generují na základě parametru `completion_percentage`, který určuje kolik procent z celkového počtu aktivit má uživatel splnit za sebou. Milníky jsou poté generovány po 7 splněních.

Výpis kódu 3.4 Generování milníků pro počet splnění aktivity za sebou

```
def create_streak_achievement(new_activity, user, completion_percentage,
                              total_instances):
    new_achievements = []
    stop = round((total_instances/100)*completion_percentage)
    for i in range(7, stop, 7):
        new_achievement = Achievement(activity=activity, created_by=user,
                                      type=STREAK, progress=0, goal=i)
        new_achievements.append(new_achievement)

    return new_achievements
```

3.3 Registrační proces s verifikací emailové adresy

Po vyplnění registračního formuláře je uživateli zaslán verifikační email. Funkce `render_to_string` vygeneruje emailovou zprávu pomocí HTML šablony, které je předán slovník se zašifrovaným id uživatele a vygenerovaným tokenem z údajů uživatele jako kontext.

Uživatelům je zobrazena zpráva ohledně stavu registrace pomocí Django Messages frameworku.

Výpis kódu 3.5 Poslání emailu pro verifikaci nového uživatele

```
class UserRegisterView(FormView):
    form_class = UserRegisterForm
    template_name = "register.html"
    success_url = reverse_lazy("index")

    def form_valid(self, form):
        try:
            with transaction.atomic():
                form = self.form_class(self.request.POST)
                user = form.save()
                user.is_active = False
                user.signup_confirmation = False
                user.save()

                current_site = get_current_site(self.request)

                subject = 'Confirm your email address'
                message = render_to_string('activation_request_register.html', {
                    'domain': current_site.domain,
                    'uid': urlsafe_base64_encode(force_bytes(user.pk)),
                    # method will generate a hash value with user related data
                    'token': account_token_generator.make_token(user),
                })
                user.email_user(subject, message, fail_silently=False)

                messages.success(self.request, "Email has been sent. Check your
                                                email for further details.")
                return redirect('account:register')

        except Exception as e:
            messages.warning(self.request,
                             "Error trying to send email. If problem persists,
                             please contact staff members.")
            return redirect('account:register')
```

Odkaz ve verifikačním emailu se namapuje na funkci `activate_registration`, které jsou předány parametry zparsované z url adresy. Následně tato funkce dekóduje uid a provede validaci tokenu s uživatelem s příslušným uid.

Výpis kódu 3.6 Dokončení registrace

```
def activate_registration(request, uidb64, token):
    try:
        uid = force_text(urlsafe_base64_decode(uidb64))
        user = User.objects.get(pk=uid)
    except (TypeError, ValueError, OverflowError, User.DoesNotExist):
        user = None

    if user is not None and account_token_generator.check_token(user, token):
        user.is_active = True
        user.signup_confirmation = True
        user.save()
        login(request, user)
        messages.success(request, "Welcome " + user.username + "!")
        return redirect('home:home_weekly')
    else:
        messages.warning(request, "Link has expired.")
        return redirect('account:register')
```

3.4 Použití AJAX

Technologii AJAX používám pro zpracování uživatelského požadavku(POST) pro aktualizaci stavu aktivity. Metoda `closest` najde nejbližšího rodiče volajícího elementu s HTML class rovnající se parametru metody. Po kliknutí na *Submit* tlačítko se najdou relevantní data a pomocí metody `ajax` se data pošlou na View, který požadavek zpracuje.

Rozhodl jsem se při úspěchu metody `ajax` znovu načíst stránku, protože na stránce odkud byl AJAX request poslán mohou být i jiné prvky používající stavy aktivit(týdenní/měsíční přehled aktivit, statistiky aktivit apod.) a musel bych také aktualizovat tyto prvky. Z tohoto důvodu je volba obnovení stránky efektivnější.

Výpis kódu 3.7 Zpracování události změny stavu aktivity

```
$(".activity-submit").click(function() {
    var metadata = $(this).closest(".sidebar-activity").find(".activity-metadata")
    var activity_pk = metadata.find("input[name='activity_pk']").val()
    var csrf_token = metadata.find("input[name='csrfmiddlewaretoken']").val()
    var update_activity_url = metadata.find("input[name='activity_url']").val()

    var complete = $(this).closest(".sidebar-activity")
                          .find(".activity-checkbox")
                          .is(":checked")
    var start_difficulty = $(this).closest(".sidebar-activity")
                                  .find(".start-difficulty-select
                                         option:selected")
                                  .val()
    var activity_difficulty = $(this).closest(".sidebar-activity")
                                     .find(".activity-difficulty-select
                                            option:selected")
                                     .val()

```

```
19      $.ajax({
20          method: "POST",
21          url: update_activity_url,
22          data: {
23              csrfmiddlewaretoken: csrf_token,
24              activity_pk: activity_pk,
25              start_difficulty: start_difficulty,
26              activity_difficulty: activity_difficulty,
27              complete: complete,
28          },
29          success: function() {
30              window.location.reload();
31          },
32          error: function(xhr){
33              $("#messages").append('<div class="alert alert-danger
34                                     text-center align-center h6" id="hideMe">
35                                     Error processing request.</div>');
36          }
37      })
38  })
```

View zvaliduje data, změní hodnoty dané aktivity na hodnoty z AJAX POST requestu, aktualizuje milníky k dané aktivitě a úroveň uživatele podle nových hodnot aktivity.

Výpis kódu 3.8 Zpracování AJAX requestu

```
1  class UpdateActivityInstanceView(LoginRequiredMixin, View):
2
3      def post(self, request, *args, **kwargs):
4          try:
5              activity_pk = request.POST.get("activity_pk")
6              complete = request.POST.get("complete")
7              start_difficulty = request.POST.get("start_difficulty")
8              activity_difficulty = request.POST.get("activity_difficulty")
9          except KeyError:
10             return HttpResponse(status=403)
11
12         activity_instance = get_object_or_404(ActivityInstance, pk=activity_pk)
13
14         if self.request.user != activity_instance.activity.created_by:
15             return HttpResponse(status=403)
16
17         if complete == "true":
18             new_activity_state = True
19         elif complete == "false":
20             new_activity_state = False
21         else:
22             return HttpResponse(status=403)
23
24         old_activity_state = activity_instance.is_complete
25         activity_instance.is_complete = new_activity_state
26         activity_instance.start_difficulty = start_difficulty
27         activity_instance.activity_difficulty = activity_difficulty
28         activity_instance.save()
29
30         if old_activity_state != new_activity_state:
31             self.request.user.evaluate_level_progress(activity_instance)
32             achievements = activity_instance.activity.achievements.all()
33             for achievement in achievements:
34                 achievement.evaluate_achievement(activity_instance)
35
36         return HttpResponse(status=200)
```

3.5 Aktualizace milníků

Aktualizace milníků probíhá pokaždé, když uživatel změní stav splnění aktivity a pošle AJAX POST request(viz. 3.4). Jednou z nedostatků Django ORM je nemožnost polymorfismu modelů(lze docílit pomocí knihovny django-polymorphic).

Milníky se v zásadě liší pouze v tom, jak se zpracuje jejich pokrok ke splnění milníku. Většina typů milníků mají stejné atributy: splněno/nesplněno, cíl a současný pokrok ke splnění cíle. Z těchto důvodu jsem se rozhodl pro jeden model a rozdělení logiky na základě typu milníku.

Výpis kódu 3.9 Metody modelu Achievement pro aktualizaci milníku

```
1
2  def evaluate_completion_achievement(self, activity_instance):
3          if activity_instance.is_complete:
4              self.progress = self.progress + 1
5          else:
6              self.progress = self.progress - 1
7
8          if self.goal == self.progress:
9              self.is_complete = True
10             self.create_notification()
11         self.save()
12
13 def evaluate_streak_achievement(self):
14     activity_instances = self.activity.activity_instances.all()
15                                                          .order_by('date')
16     streak = 0
17     best_streak = 0
18     for activity_instance in activity_instances:
19         if activity_instance.is_complete:
20             streak = streak + 1
21             if streak > best_streak:
22                 best_streak = streak
23             if streak == self.goal:
24                 self.is_complete = True
25                 self.create_notification()
26                 break
27         else:
28             streak = 0
29     self.progress = best_streak
30     self.save()
31
32 def create_notification(self):
33     if self.type == NUMBER_OF_COMPLETION:
34         Notification.objects.create(text=f"Completed {self.activity.name}
35                                     {self.goal} times.",
36                                     created_by=self.created_by)
37     elif self.type == HOT_STREAK:
38         Notification.objects.create(text=f"Completed {self.activity.name}
39                                           {self.goal} times in a row.",
40                                           created_by=self.created_by)
41     else:
42         return
43
44 def evaluate_achievement(self, activity_instance):
45     if self.is_complete:
46         return
47     if self.type == NUMBER_OF_COMPLETION:
48         return self.evaluate_completion_achievement(activity_instance)
49     elif self.type == HOT_STREAK:
50         return self.evaluate_streak_achievement()
```

3.6 Zpracování statistik

View komponenty, zpracovávající požadavky pro zobrazení statistik(měsíční/roční/celkové), zavolají funkci `get_overall_statistics` a předají datový interval ke zpracování statistik. Tato funkce zavolá funkce pro zpracování jednotlivých druhů statistik.

Výpis kódu 3.10 Zpracování statistik

```
def get_overall_statistics(user, date_format, start_date=None, end_date=None):
    date_today = datetime.today()
    if not (start_date and end_date):
        instances = ActivityInstance.objects.filter(activity__created_by=user,
                                                    date__lte=date_today)
                                            .order_by("date")

    elif date_today <= end_date:
        instances = ActivityInstance.objects.filter(date__gte=start_date,
                                                    date__lte=date_today,
                                                    activity__created_by=user)
                                            .order_by("date")
    else:
        instances = ActivityInstance.objects.filter(date__gte=start_date,
                                                    date__lte=end_date,
                                                    activity__created_by=user)
                                            .order_by("date")

    day_dict = get_statistics_by_day(date_format, instances)
    activity_dict = get_statistics_by_activities(instances)
    activity_composition_dict = get_activity_composition(instances)
    start_difficulty_dict = get_average_start_difficulty(instances)
    difficulty_dict = get_average_difficulty(instances)

    return day_dict, activity_dict, activity_composition_dict,
           start_activity_difficulty_dict, activity_difficulty_dict
```

Funkce pro zpracování jednotlivých statistik vytváří slovník. Ten má jako klíče prvky, které se zobrazí na ose x a jednotlivé klíče mají jako hodnotu prvky, které se zobrazí na ose y(pro souřadnice [x, y] platí x=x, y=dict[x]). Seřazenost instancí aktivit zaručuje, že se hodnoty na ose x vygenerují seřazené podle data.

Hodnoty slovníku jsou objekty třídy Statistics, ve kterých jsou uloženy informace pro výpočet statistik.

Výpis kódu 3.11 Zpracování statistiky podle datového formátu

```
class Statistics:
    completed = 0
    total = 0
    def calculate_percentage(self):
        if self.total == 0 or self.completed == 0:
            return 0
        else:
            return round(self.completed / (self.total / 100))

    def calculate_average(self):
        if self.total == 0 or self.completed == 0:
            return 0
        else:
            return round(self.completed / self.total, 1)
```

```
15
16 def get_statistics_by_day(date_format, activity_instances):
17     day_dict = {}
18     for instance in activity_instances:
19         instance_format = instance.date.strftime(date_format)
20
21         if not day_dict.get(instance_format):
22             day_dict[instance_format] = Statistics()
23
24         if instance.is_complete:
25             day_dict[instance_format].completed += 1
26
27         day_dict[instance_format].total += 1
28
29     return day_dict
```

Jak si můžete všimnout z výpisu kódu 3.10, volání funkcí pro výpočet jednotlivých druhů statistik v zásadě dělají to samé(prochází všechny instance a vypočítávají statistiky). Nemusel bych to celé rozdělit do funkcí a mít výpočet v jednom velkém cyklu, který prochází všechny instance.

Nicméně, jedná se pouze o zmenšení multiplikativní konstanty(časová složitost algoritmu je stále lineární), a proto jsem se rozhodl pro rozdělení funkcionality do menších funkcí pro udržitelnost kódu.

3.7 Zobrazení grafů pomocí Graph.js

Lze využít funkcionality značkovacího jazyka Django frameworku(DTL) a vygenerovat hodnoty(získané z 3.10) pro grafy v HTML šabloně. Je nutné tyto hodnoty vygenerovat v HTML šabloně, protože DTL neumí pracovat se soubory obsahující pouze JavaScript kód(soubory s příponou .js).

Výpis kódu 3.12 Příprava dat pro grafy pomocí DTL

```
1     <script>
2     day_dict_labels = [];
3     day_dict_values = [];
4     {% for key, value in day_dict.items %}
5         day_dict_labels.push( {{key}} );
6         day_dict_values.push( {{value.calculate_percentage}} );
7     {% endfor %}
8     </script>
```

Vygenerované hodnoty lze použít podle dokumentace Graph.js a zobrazit grafy. [24]

Výpis kódu 3.13 Zobrazení grafu pomocí Graph.js

```
1 ctx = document.getElementById('day_dict_chart').getContext('2d');
2 myChart = new Chart(ctx, {
3     type: 'bar',
4     data: {
5         labels: day_dict_labels,
6         datasets: [{
7             label: 'Success rate by weekdays',
8             data: day_dict_values,
9             backgroundColor: ['rgba(50, 205, 50, 0.2)'],
10            borderColor: ['rgba(50, 205, 50, 1)'],
```

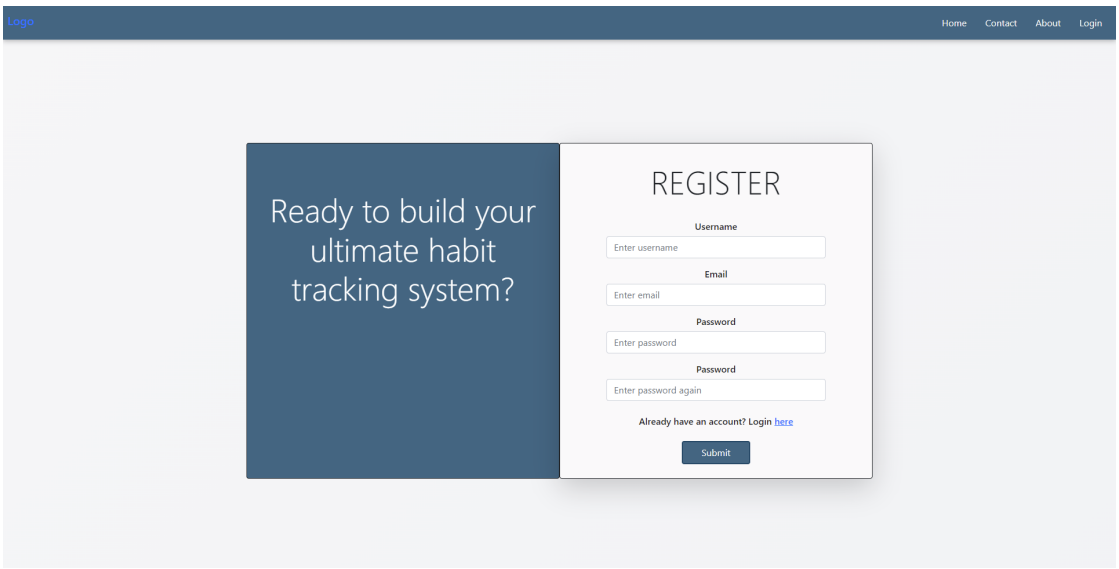
```
            fill: false,
            tension: 0.1,
            borderWidth: 1
        }]
    },
    options: {
        scales: {
            y: {
                suggestedMax: 100,
                beginAtZero: true,
                ticks: {
                    // Include a dollar sign in the ticks
                    callback: function(value, index, ticks) {
                        return value + '%';
                    }
                }
            }
        }
    }
});
```

3.8 Uživatelské rozhraní

Pro implementaci uživatelského jsem použil HTML, CSS, JavaScript a Bootstrap. Dynamické prvky se generují pomocí značkovacího jazyka Django frameworku. Návrhy všech stránek aplikace je možné si prohlédnout v příloze A.

3.8.1 Domovská stránka

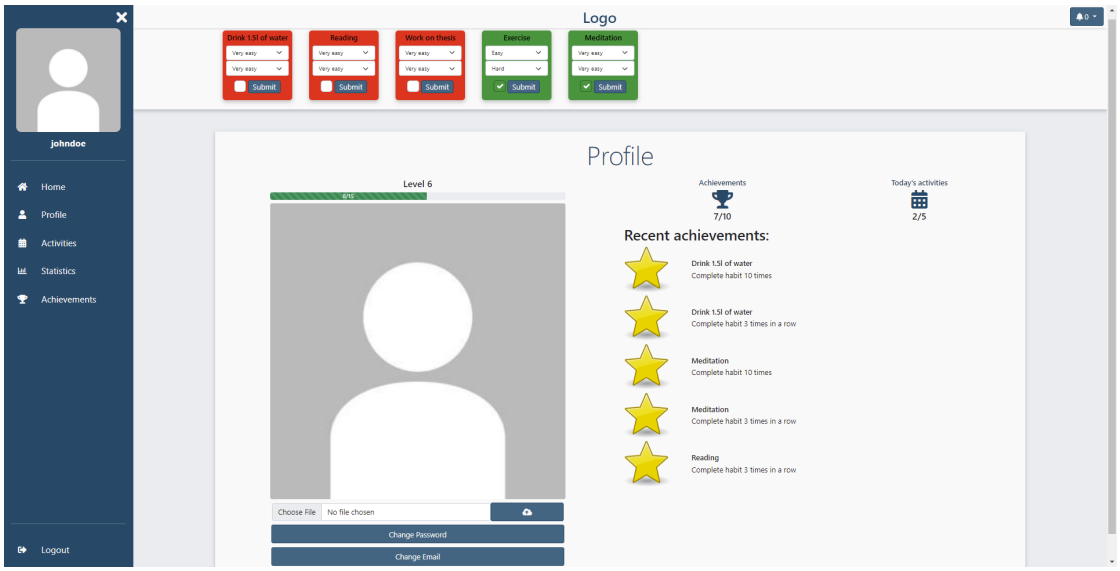
Snažil jsem se držet podle návrhu(obrázek 2.3). Pro vyřešení responsivity jsem využil grid systému Bootstrap knihovny. Pokud je šířka obrazovky malá, tak se prvky zarovnají nad sebe.



Obrázek 3.1 Domovská stránka

3.8.2 Profilová sekce

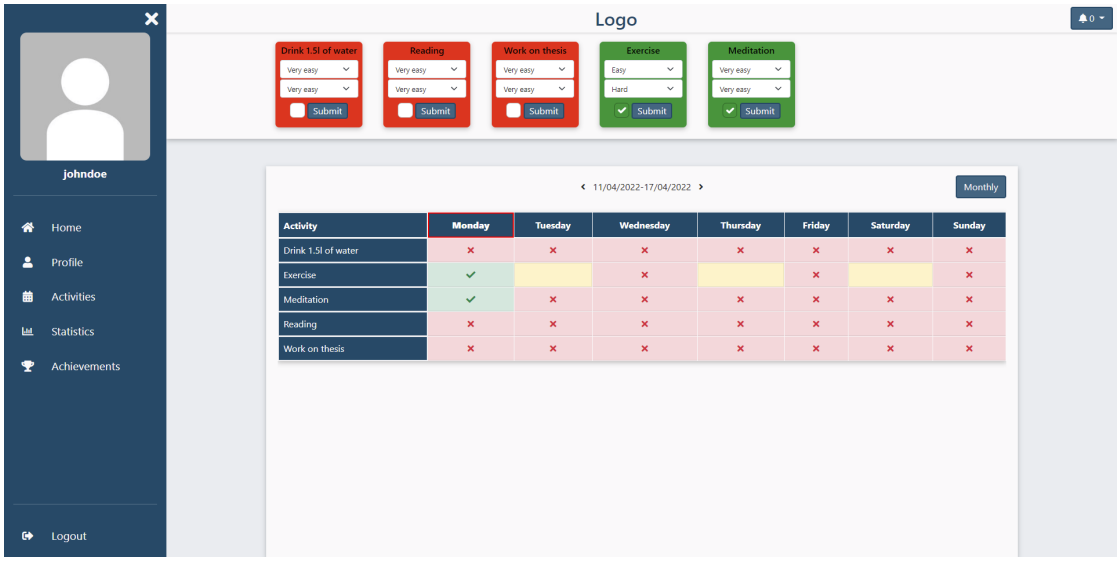
Pro zobrazení současného pokroku do další úrovně uživatele jsem použil Bootstrap komponentu Progress. Responsivita této stránky je také řešena pomocí grid systému.



Obrázek 3.2 Profilová sekce

3.8.3 Tabulka s přehledem stavů aktivit

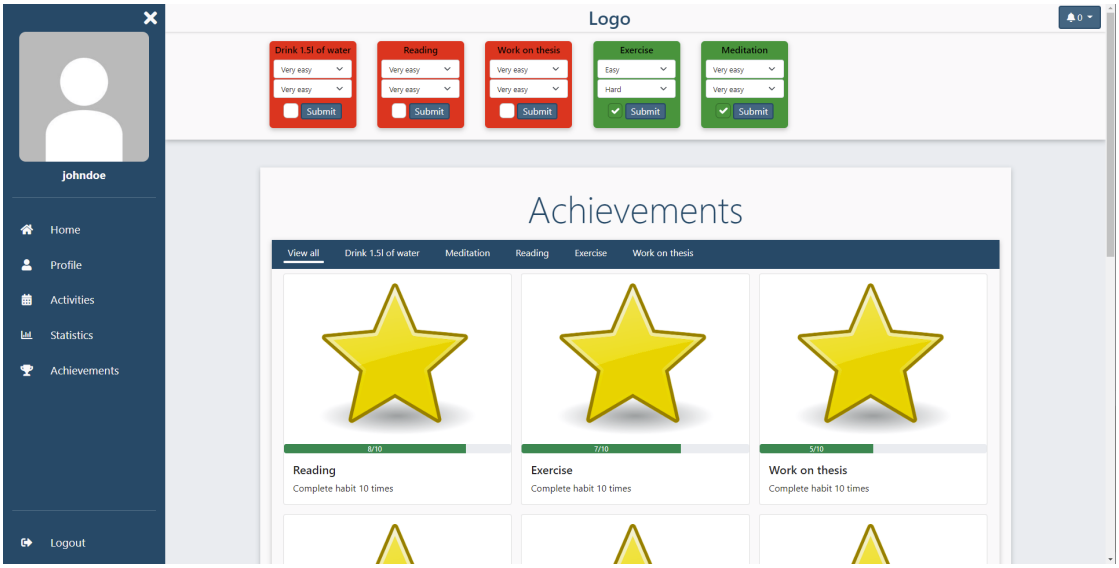
Responsivitu tabulky řeším pomocí Bootstrap tabulky. Kliknutím na jednotlivé buňky tabulky se otevře Bootstrap modální okno pro editaci stavu dané aktivity.



Obrázek 3.3 Přehled stavů aktivit

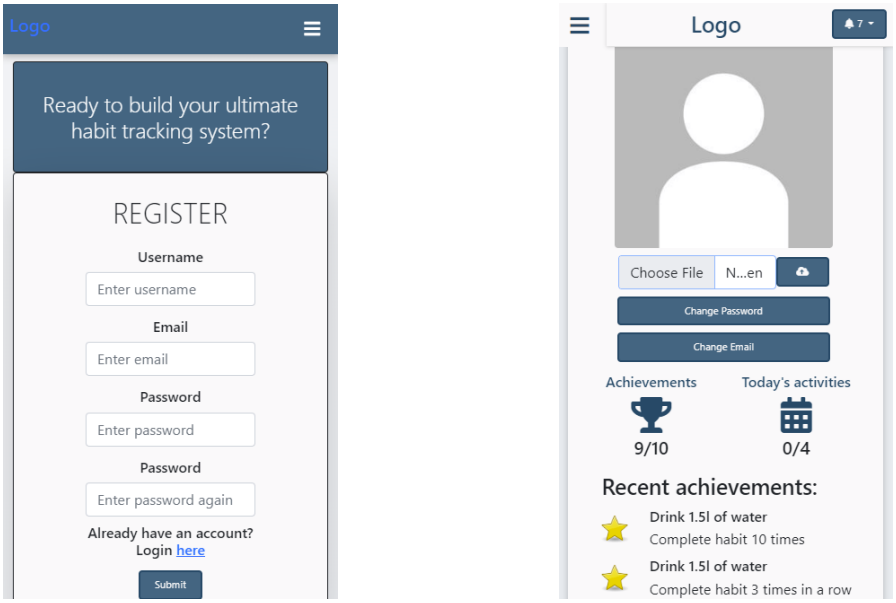
3.8.4 Přehled milníků

Pro implementaci jednotlivých milníků jsem použil Bootstrap komponentu Card. Responsivita jednotlivých milníků je také řešena pomocí Bootstrap grid systému.

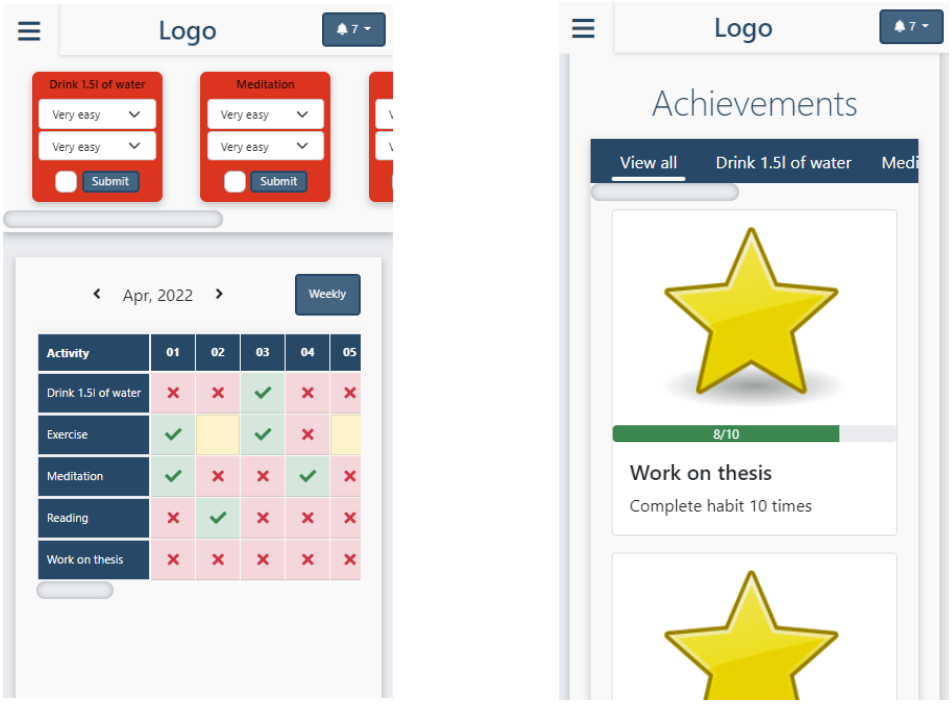


Obrázek 3.4 Přehled milníků

3.8.5 Zobrazení na malých zařízení



Obrázek 3.5 Domovská stránka a profilová sekce na malých zařízení



Obrázek 3.6 Přehled aktivit a milníků na malých zařízení

Kapitola 4
Testování

Po dokončení implementace aplikace do použitelného stavu je nutné produkt otestovat. Testování je jednou z nejdůležitějších fází po implementaci před nasazením aplikace do produkce.
V této kapitole se zaměřím na testování ve frameworku Django a následně okomentuji a zhodnotím výsledky uživatelských testů použitelnosti.

4.1 Testování

Automatizované testování je extrémně užitečný nástroj na odchycení a vyřešení chyb pro moderní webové aplikace. Pokud provádíme refactoring kódu, testy dokážou zajistit, aby změny neočekávaně neovlivnily chování testované aplikace.

Testování webové aplikace je složitý úkol, protože webová aplikace se skládá z několika vrstev logiky, od zpracování požadavků na úrovni HTTP, přes ověřování a zpracování formulářů až po vykreslování šablon. S frameworkem Django a různými nástroji lze simulovat požadavky, vytvářet a pracovat s testovacími daty, kontrolovat výstup aplikace a obecně ověřovat, jestli kód vykonává to, co by měl. [27]

Během implementace jsem se soustředil na testování kritických funkcí, které zpracovávají data k uložení do databáze. U každého TestCase jsem nejprve v metodě `setUp` vytvořil testovací data(Django automaticky vytváří/maže testovací databázi) a následně tyto data používal v jednotlivých testech. Pro evaluaci výsledku s očekávanou hodnotou lze použít např. metoda `assertEqual` třídy `TestCase`.

Příklad testu funkcí z výpisů kódů 3.3 a 3.4:

Výpis kódu 4.1 Test funkcí pro generování milníků

```
1  from django.test import TestCase
2
3  class AchievementTestCase(TestCase):
4
5      def setUp(self):
6          user1 = User.objects.create(username="user1")
7          Activity.objects.create(name="activity1", created_by=user1,
8                                  start_date=datetime.datetime(2022, 4, 1),
9                                  end_date=datetime.datetime(2022, 4, 30))
10
```

```
11      def test_create_completion_achievement(self):
12          total_instances = 30
13          num_of_equal_parts = 4
14          user1 = User.objects.get(username="user1")
15          activity1 = Activity.objects.get(name="activity1")
16
17          new_achievements = create_completion_achievement(activity1,
18                                                           user1,
19                                                           num_of_equal_parts,
20                                                           total_instances)
21
22          self.assertEqual(len(new_achievements), num_of_equal_parts)
23          self.assertEqual(new_achievements[0].goal, 8)
24          self.assertEqual(new_achievements[1].goal, 15)
25          self.assertEqual(new_achievements[2].goal, 22)
26          self.assertEqual(new_achievements[3].goal, 30)
27
28      def test_create_streak_achievement(self):
29          total_instances = 30
30          completion_percentage = 50
31          user1 = User.objects.get(username="user1")
32          activity1 = Activity.objects.get(name="activity1")
33          new_achievements = create_streak_achievement(activity1,
34                                                       user1,
35                                                       completion_percentage,
36                                                       total_instances)
37          self.assertEqual(len(new_achievements), 2)
38          self.assertEqual(new_achievements[0].goal, 7)
39          self.assertEqual(new_achievements[1].goal, 14)
```

Dalším užitečným nástrojem je zabudované prostředí pro vývojáře dostupné ve většině moderních webových prohlížečích. Díky tomuto nástroji jsem mohl otestovat svůj responsivní design na různých rozlišení. Aplikaci jsem pravidelně testoval během vývoje tím, že jsem jí sám využíval ke sledování osobního rozvoje.

4.2 Testování použitelnosti

Testování použitelnosti je praktika, která testuje funkčnost a použitelnost aplikace. Je založena na pozorování uživatelů při vykonávání úkolů, související s použitím aplikace. Na základě zpětné vazby, která vznikla během testování uživateli, lze usoudit, jak je výsledný softwarový produkt použitelný a jakou reakci budou mít uživatelé běžném používání.

4.2.1 Průběh testování

Testování se zúčastní 3 lidé s různými znalostmi o vyvíjené aplikaci. Jedna osoba o aplikaci neslyšela, druhá byla seznámena se zadáním a třetí osoba viděla vývoj aplikace a je seznámena s hlavními funkcionalitami. Před každým testem moderátor připraví aplikaci do stavu podle předpokladů testovacího scénáře. Následně se uživatelé pokusí o splnění testovacího scénáře a moderátor si zapisuje výsledky a popřípadě zpětnou vazbu od uživatelů. Testovací scénáře jsou zaměřeny na nejběžnější činnosti vykonávaných uživateli. Všechny testovací scénáře je možné si prohlédnout v příloze B.

Příklad testovacího scénáře:

TS1	**Registrace uživatele**
Kontext:	Uživatel se dozvěděl o aplikaci a chtěl by se zaregistrovat.
Předpoklady:	Moderátor připraví nezaregistrovanou emailovou adresu, pomocí které se může uživatel zaregistrovat.
Úspěch:	Uživatel se úspěšně zaregistroval, ověřil emailovou adresu a nachází se v rozhraní přihlášeného uživatele.
Postup:	1. Do prohlížeče zadejte adresu domovské stránky a vyčkejte až se Vám zobrazí domovská stránka s registračním formulářem. 2. Nalezněte registrační formulář a vyplňte ho s poskytnutými údaji, které Vám předá moderátor. 3. Po vyplnění formuláře klikněte na tlačítko *Submit*. 4. Přihlašte se do zadané emailové adresy a ověřte emailovou adresu pomocí odkazu ve verifikačním emailu. 5. Po stisknutí na verifikační odkaz se nacházíte v rozhraní přihlášeného uživatele.

4.2.2 Výsledky testování

První osoba, která o aplikaci vůbec nevěděla, měla problém s verifikací emailové adresy, protože si nestihla přečíst zprávu, která oznámila, že registrace proběhla úspěšně a je potřeba si verifikovat zadanou emailovou adresu. Poté, co se seznámila se systémem pro verifikaci emailové adresy, splnila další testovací scénáře bez problému. Druhá osoba a třetí osoba splnily všechny testovací scénáře bez větších obtíží.

Obecně byla aplikace hodnocena pozitivně. Ucelenost návrhu UI a použití běžných designů webových aplikací pomohlo ke snadné orientaci v aplikaci. Nicméně druhá osoba podotkla, že by bylo dobré aplikaci přidat různé žebříčky anebo jiné způsoby srovnávání s ostatními.

Jako reakce na problém, na který narazila první osoba, je délka zobrazení oznamovací zprávy prodloužena.

Kapitola 5

Přínosy a možná vylepšení aplikace

Po dokončení fází analýzy, návrhu a implementace se vytvoří dobrý náhled do domény a z těchto poznatků lze navrhnout možná vylepšení aplikace do budoucnosti.
V této kapitole zhodnotím možnosti rozšíření, flexibilitu a interoperabilitu výsledné aplikace. Shrnu přínosy aplikace a zaměřím se na rozšíření, které by mohli být přínosné v budoucnosti. Tato vylepšení vzešla během analýzy a implementaci této práce.

5.1 Přínosy aplikace

Aplikace poskytuje registraci, kompletní autentizaci uživatele, přizpůsobení profilu uživatele a možnost definovat vlastní aktivity. U každé aktivity je možné sledovat stav splněno/nesplněno, obtížnost začít aktivitu a obtížnost aktivity. K definovaným aktivitám jsou uživateli poskytnuty různé formy sledování pokroku v dodržování těchto aktivit jako jsou například:

- Zobrazení tabulky s přehledem stavů aktivit v různých časových intervalech.
- Zobrazení celkových statistik aktivit v různých časových intervalech.
- Zobrazení celkových statistik aktivity.

Pro zvýšení úspěšnosti dodržování aktivit jsou uživatelům poskytnuty různé herní prvky(systém úrovní pro uživatele a milníky k aktivitám), které působí motivačně a přidávají zábavné elementy aplikaci.

5.2 Možnosti rozšíření a flexibilita

Aplikace je vyvíjena jako webová aplikace pomocí architektury MVT, kterou framework Django dodržuje a programátoři se tak nemusí starat o organizaci kódu, jelikož framework generuje doporučovanou strukturu projektu. Rozšíření funkcionalit je tedy jednoduché, jedná se pouze o vytvoření nové části aplikace pomocí příkazu, který vygeneruje strukturu nové části aplikace.

Během implementace jsem se snažil dekomponovat rozsáhlé funkce do menších logických funkcí. Například přidání nového typu statistik lze napsáním nové funkce pro zpracování dat k zobrazení dané statistiky. Tyto funkce jsou poté znovupoužitelné a zároveň lépe udržovatelné. Rozšíření sledovaných hodnot u aktivity lze pomocí nového atributu v modelu aktivity. Přidání nového typu milníku lze docílit pomocí rozšíření typů milníků v modelu a napsání logiky evaluace.

Zaměřením se na responsivitu návrhu je aplikace použitelná na jakémkoliv zařízení s internetovým prohlížečem a zobrazitelná na nejběžnějších rozlišení.

Django poskytuje různé možnosti pro výkonnostní rozšíření. V konfiguračním souboru si lze nastavit životnost spojení do databáze pomocí `CONN_MAX_AGE`. Také je možné v konfiguračním souboru odstranit nepotřebný Middleware. Pro další škálování je také možné si nastavit cache systém podle Django dokumentace [28].

5.3 Interoperabilita

Pokud je potřeba komunikace s ostatními systémy, lze to docílit napsáním příslušných View komponent, které vrací data ve formátech vhodných pro komunikaci přes internet. Django poskytuje konvertory dat z databáze do nejběžnějších formátů(JSON, XML apod.) pro komunikaci přes internet. Vytvořili bychom API aplikace. Interoperabilitu jsem bral v potaz během vývoje a snažil jsem se psát kód ve funkcích pro snadnost znovupoužitelnosti ve View komponentách, které by zpracovávaly volání API aplikace.

5.4 Možná rozšíření a vylepšení aplikace

Sociální síť

Aplikaci lze rozšířit přidáním funkcionalit typických u sociálních sítí:

- **Seznam přátel:** možnost přidat si uživatele a sledovat jejich milníky, stavy aktivit apod.
- **Fórum:** možnost vytvořit diskuzní téma a ostatní uživatelé by se mohli zapojit do konverzace pomocí komentářů.
- **Zprávy:** možnost posílání soukromých zpráv mezi uživateli.

Rozšíření prvků gamifikace

Podle analýzy gamifikace existují další prvky, které jsem ve svojí implementaci nepřidal, konkrétně se jedná o následující prvky:

- **Odměny:** uživatel si může definovat vlastní odměny k aktivitám.
- **Milníky:** rozšíření typů milníků(např. počet přihlášení, dosažení určité úrovně apod.).
- **Žebříčky:** přidání žebříčku úrovní uživatelů, měsíční úspěšnosti mezi uživateli nebo přáteli.

Rozšíření systému notifikací

Uživatelům by mohla být nabídnuta možnost si zvolit jestli chtějí, aby jim byly posílány SMS nebo emailové zprávy, pokud za daný den nesplní aktivitu.

U jednotlivých aktivit by byla možnost si zaškrtnout, aby se daná aktivita nepočítala jako nesplněná, a tím se neposlala zpráva.

Kategorizace uživatelů

Pokud je potřeba přidat prémiové funkcionality nebo rozdělit aplikaci do volně přístupných a prémiových funkcionalit, bude potřeba rozdělit uživatele podle jejich druhu předplatného a na základě toho omezit přístup k funkcionalitám.

Kategorizace aktivit

K jednotlivým aktivitám by mohla být přiřazena kategorie, která by byla využita jako další typ statistik podle jednotlivých kategorií.

Design aplikace

Aplikace v současném stavu vypadá celkem jednoduše. Během předělání designu lze změnit návrh komponent, barevné schéma(změna CSS proměnných, které definují barvy) nebo přidat vhodné nerušivé pozadí.

Generování aktivit

Současný způsob generování aktivit má svoje klady i zápory. Alternativní způsob může být generování dalších aktivit týdně/měsíčně, dokud uživatel nezvolí možnost skončení aktivity. Tato metoda si vyžaduje určité schopnosti se skriptováním, konkrétně automatické spouštění skriptu, který dané aktivity vytvoří v nastavených časových intervalech.

Tento způsob vyžaduje také změnu logiky generování milníků k aktivitám. Jedním ze způsobů může být generování milníků společně s aktivitami pomocí automaticky se spouštějícího se skriptu.

Autentizace

U registračních a přihlašovacích formulářů by bylo dobré přidat captcha(např. implementace poskytnuta firmou Google) proti útokům botů.

Závěr

Cílem bakalářské práce bylo provést analýzu domény, sestavení funkčních a nefunkčních požadavků, návrh a implementace webové aplikace zaměřené na podporu vytvoření nebo odstranění zvyků pomocí metody habit tracking.

V první kapitole bylo hlavním cílem se seznámit s doménou, analýza existujících řešení a identifikovat silné a slabé stránky těchto řešení. Ze získaných poznatků bylo možné sestavit funkční a nefunkční požadavky. Tyto informace posloužily k následnému vytvoření případů užití, které blíže definovaly funkcionality aplikace.

Následovala fáze návrhu, ve které byly vybrány vhodné technologie pro usnadnění implementace aplikace. Vybráním frameworku Django mi usnadnilo práci s návrhem architektury, protože framework automaticky generuje soubory a rozděluje strukturu aplikace do jednotlivých komponent podle návrhového vzoru MVT.

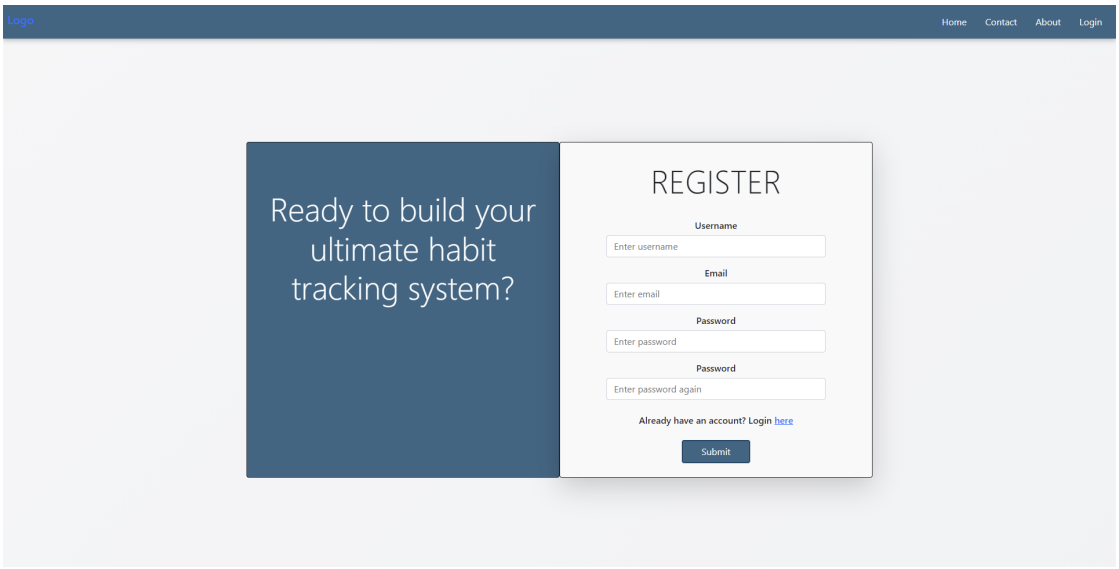
Během implementační fáze se mi dobře pracovalo s technologií Django díky předešlým znalostem ze školního projektu a práce. Tvorba responsivního uživatelského rozhraní byla značně jednodušší díky knihovně Bootstrap, která nabídla mnoho vhodných komponent pro vyřešení nejčastějších překážek při vytváření uživatelského rozhraní.

Poslední fází bylo testování aplikace, při které jsem se dozvěděl mnoho informací o použitelnosti aplikace a podkladů pro vylepšení aplikace v budoucnosti. Bylo to poprvé, kdy jsem viděl aplikaci v rukou jiného člověka a mohl jsem si představit, jak by aplikace mohla být využívána různými uživateli.

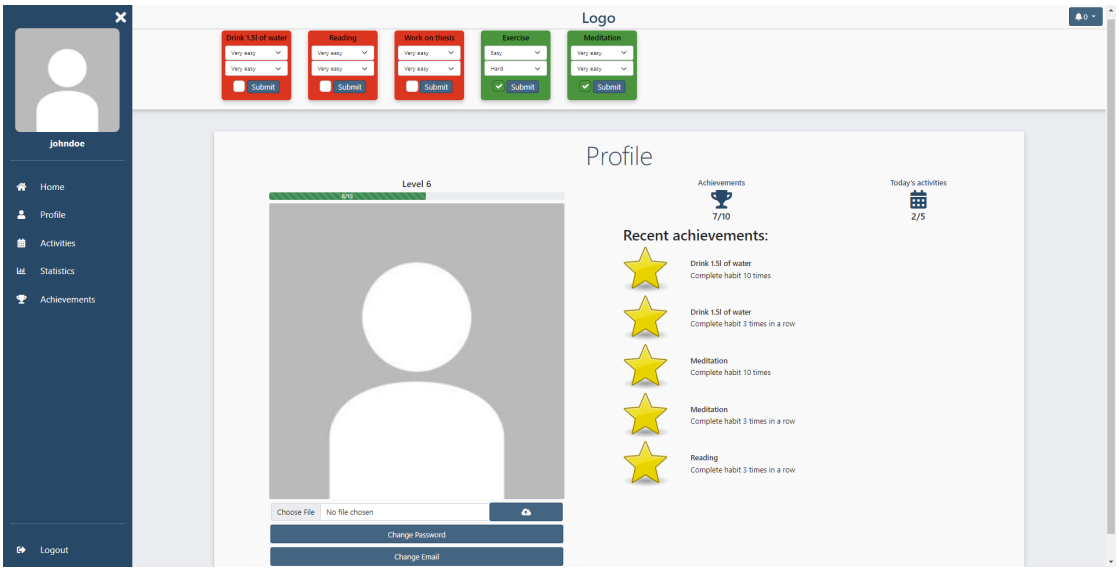
Při vývoji a psaní této práce jsem si mohl vyzkoušet všechny fáze vývoje softwaru. Z tohoto důvodu jsem blíže pochopil různé nuance vývoje softwaru, které mi pomůžou v budoucích softwarových projektech. Všechny body zadání byly zohledněny a zahrnuty do práce. Výsledkem je aplikace pro podporu a sledování osobního rozvoje.

Příloha A

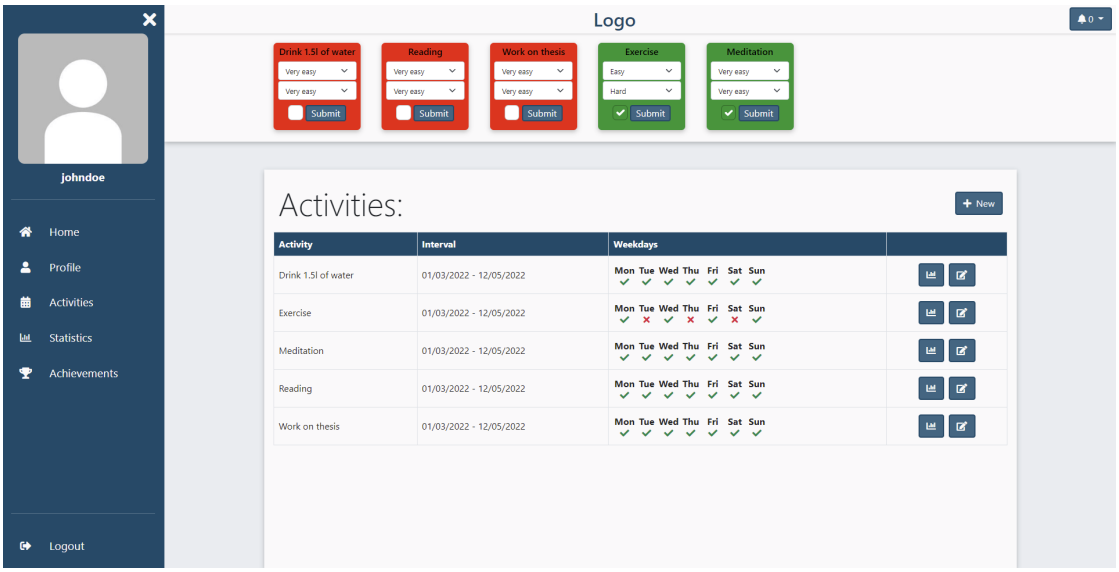
Výsledná aplikace



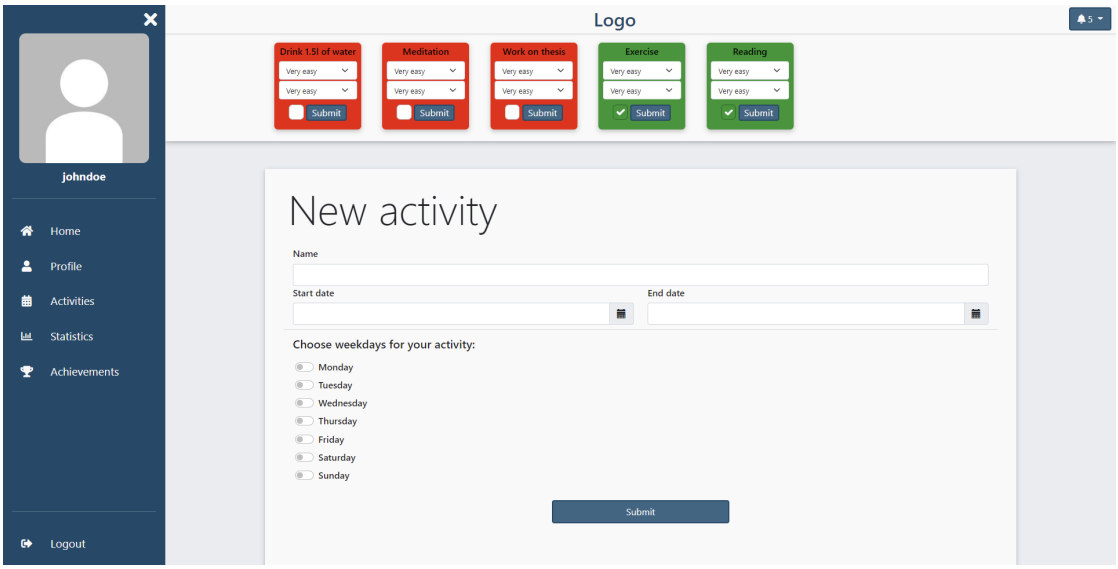
Obrázek A.1 Domovská stránka



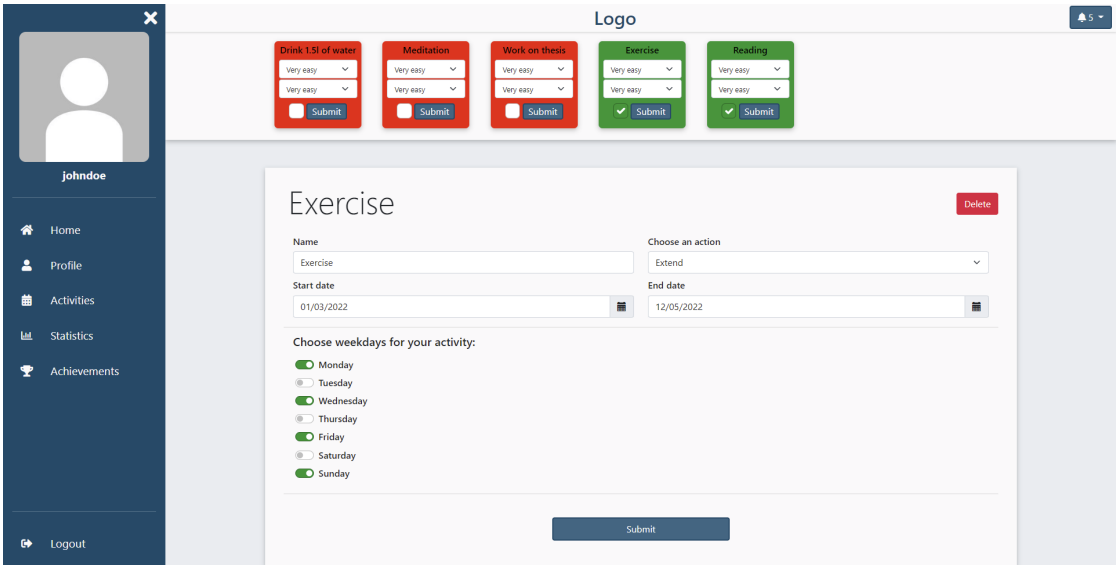
Obrázek A.2 Profilová sekce



Obrázek A.3 Přehled existujících aktivit

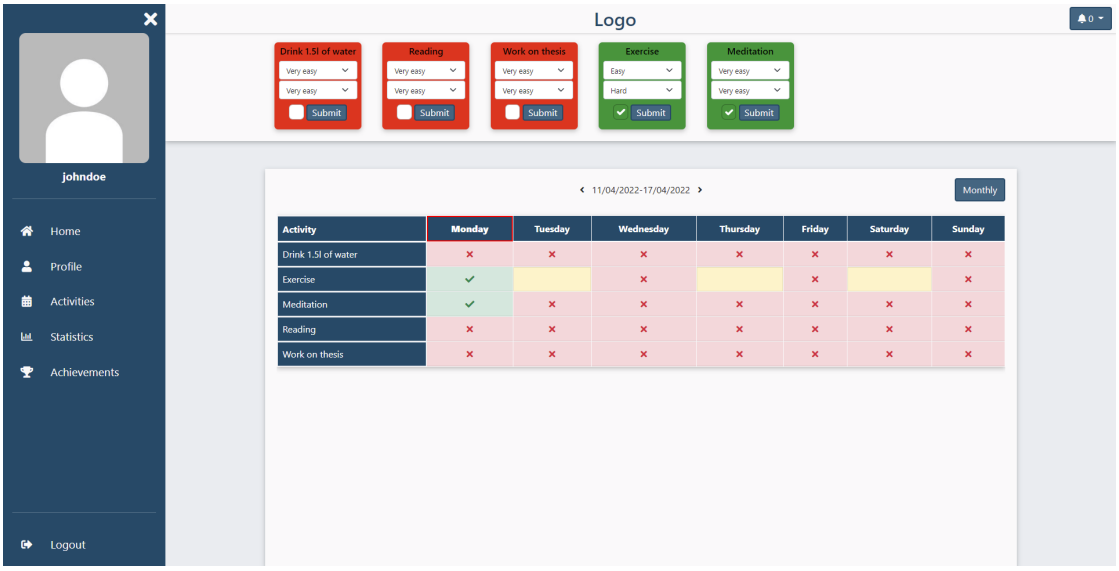


Obrázek A.4 Formulář pro vytvoření aktivity

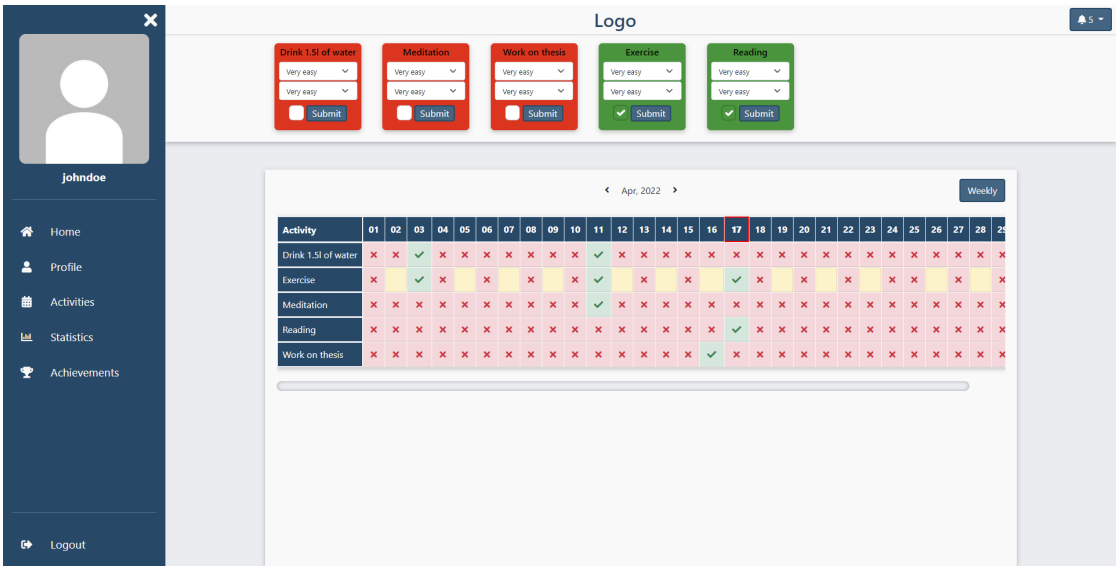


Obrázek A.5 Formulář pro editaci aktivity

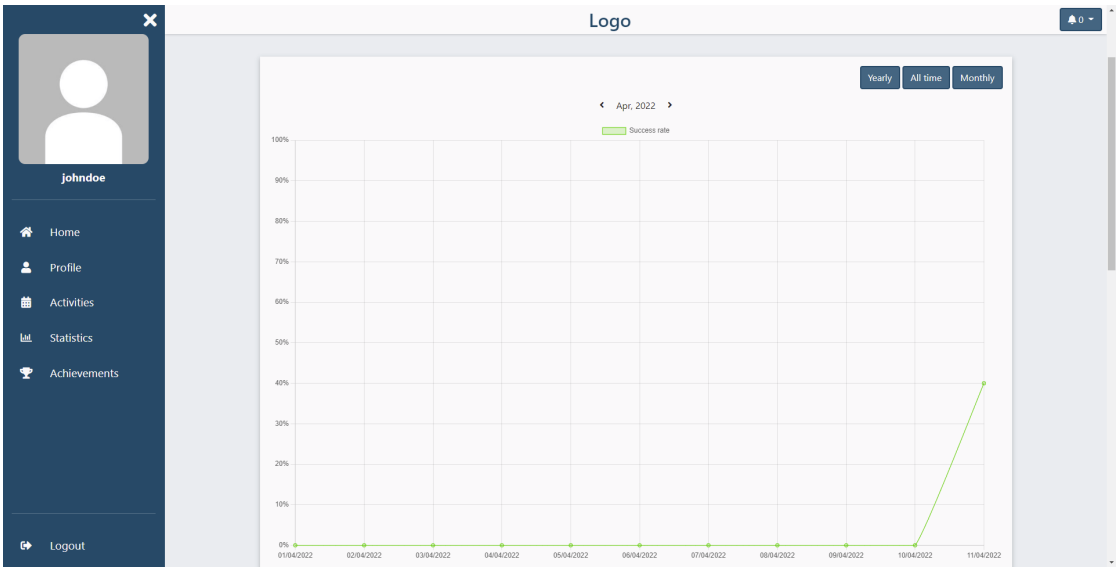
Obrázek A.6 Statistiky aktivity



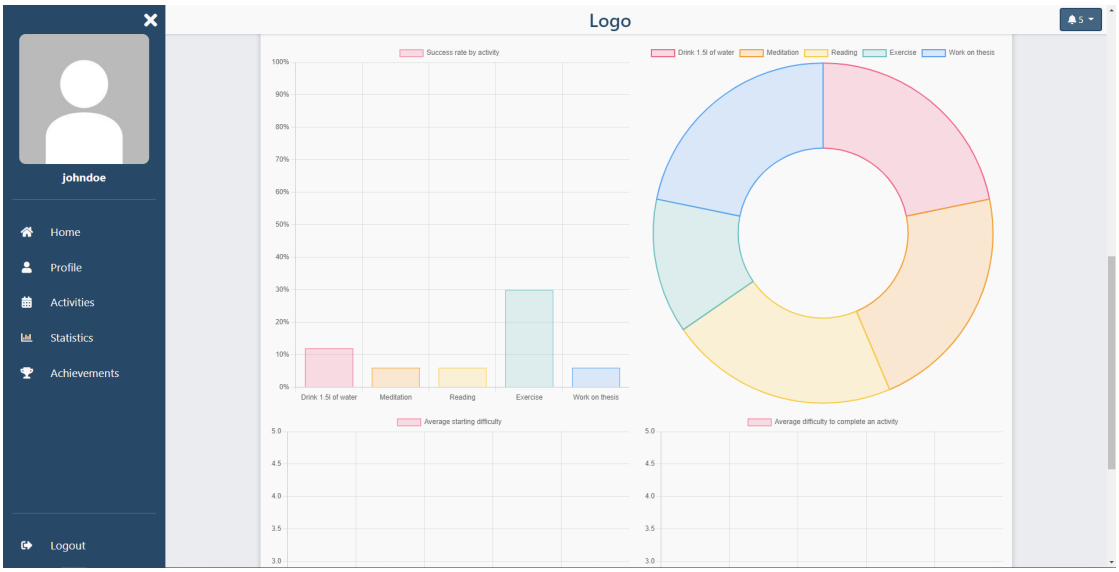
Obrázek A.7 Přehled stavů aktivit - týdenní



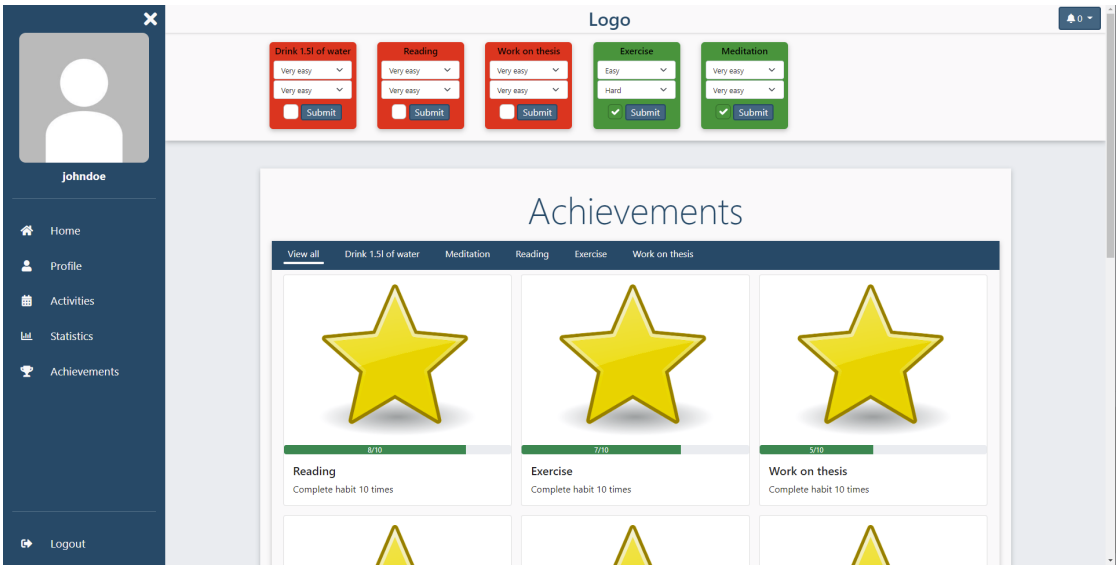
Obrázek A.8 Přehled stavů aktivit - měsíční



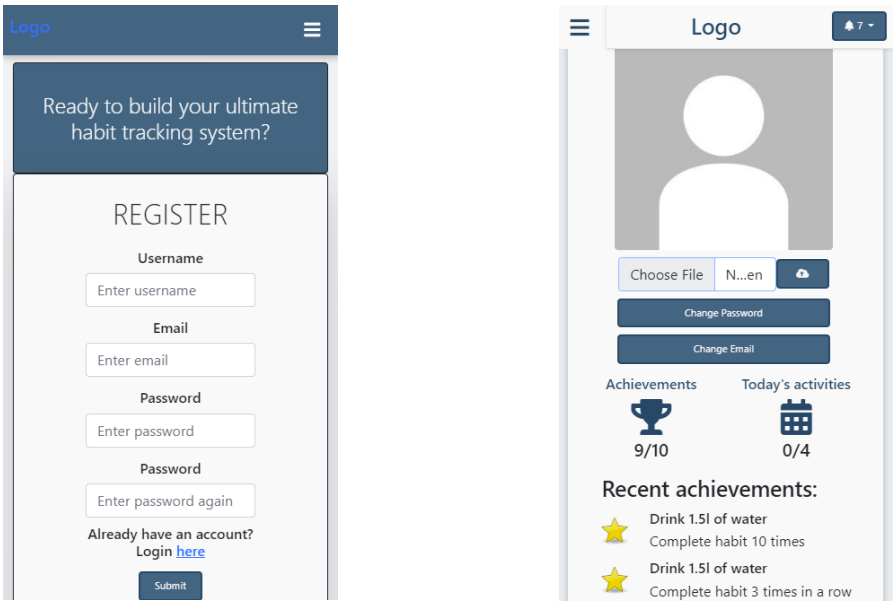
Obrázek A.9 Celkové statistiky aktivit A



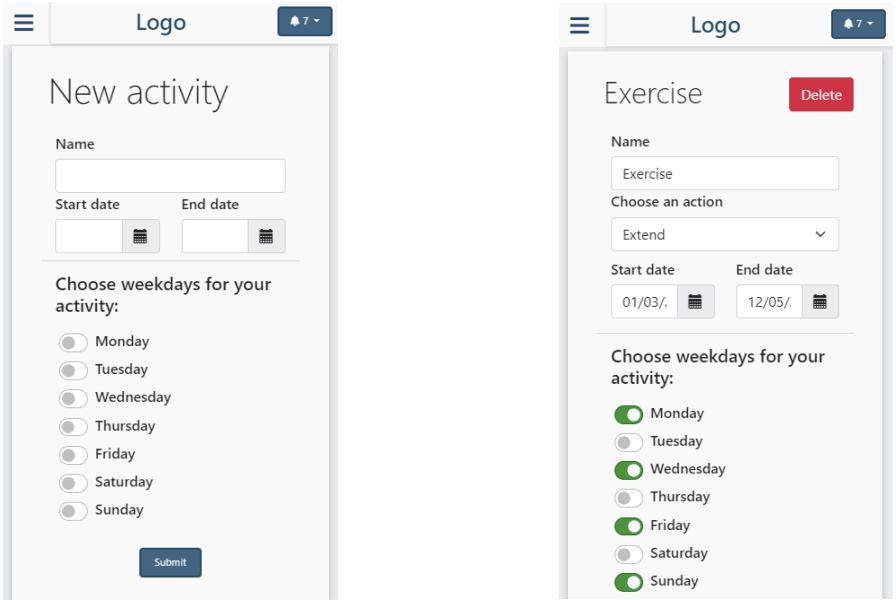
Obrázek A.10 Celkové statistiky aktivit B



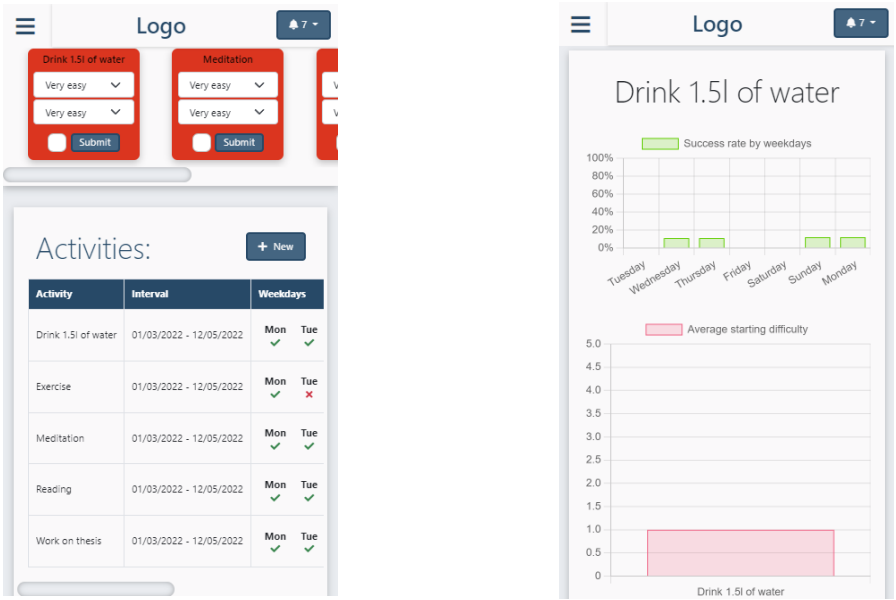
Obrázek A.11 Přehled milníků



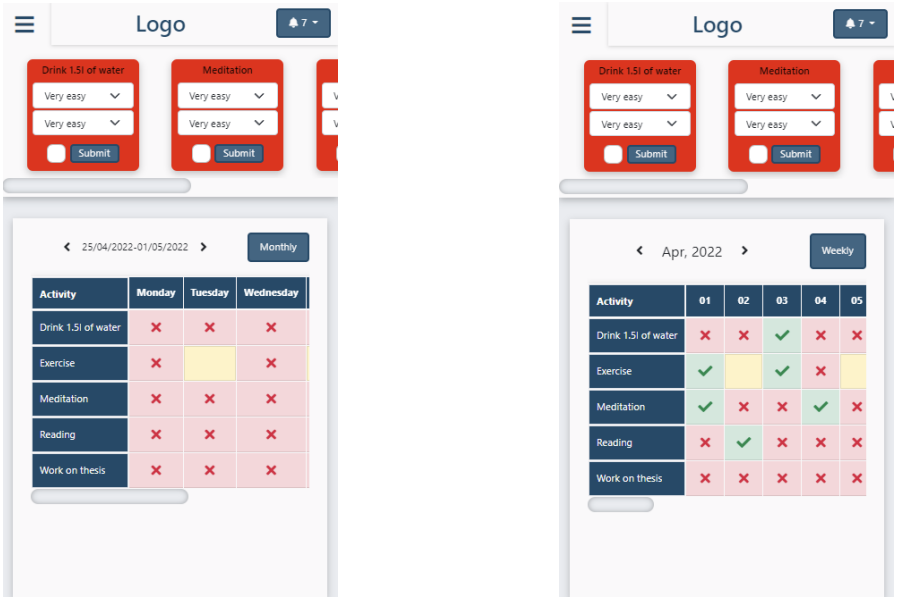
Obrázek A.12 Domovská stránka a profilová sekce na malých zařízení



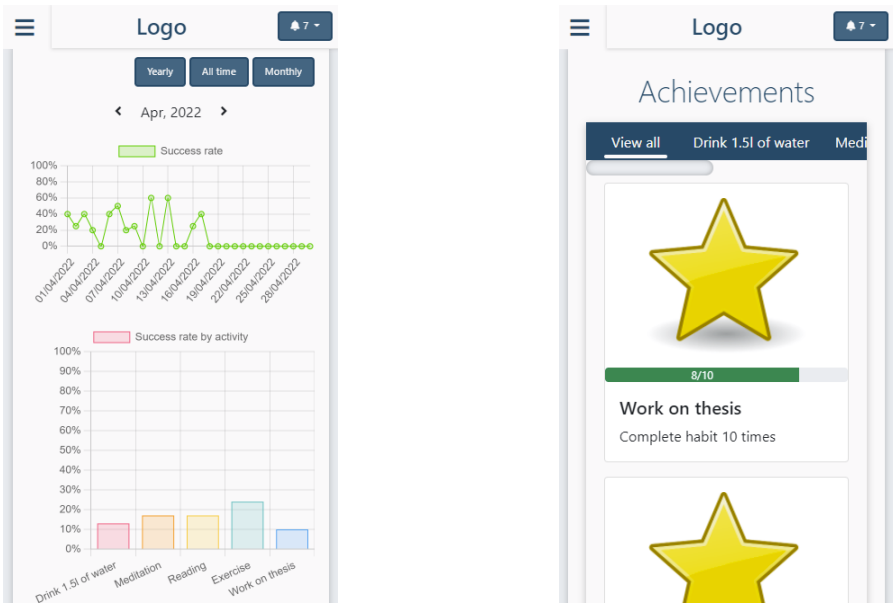
Obrázek A.13 Formuláře vytvoření a editace aktivity na malých zařízení



Obrázek A.14 Přehled aktivit a statistiky aktivity na malých zařízení



Obrázek A.15 Přehled stavů aktivit týdenní/měsíční na malých zařízení



Obrázek A.16 Přehled statistik aktivit a milníků na malých zařízení

Příloha B

Testovací scénáře

B.1 Registrace uživatele

TS1	**Registrace uživatele**
Kontext:	Uživatel se dozvěděl o aplikaci a chtěl by se zaregistrovat.
Předpoklady:	Moderátor připraví nezaregistrovanou emailovou adresu, pomocí které se může uživatel zaregistrovat.
Úspěch:	Uživatel se úspěšně zaregistroval, ověřil emailovou adresu a nachází se v rozhraní přihlášeného uživatele.
Postup:	1. Do prohlížeče zadejte adresu domovské stránky a vyčkejte až se Vám zobrazí domovská stránka s registračním formulářem. 2. Nalezněte registrační formulář a vyplňte ho s poskytnutými údaji, které Vám předá moderátor. 3. Po vyplnění formuláře klikněte na tlačítko *Submit*. 4. Přihlašte se do zadané emailové adresy a ověřte emailovou adresu pomocí odkazu ve verifikačním emailu. 5. Po stisknutí na verifikační odkaz se nacházíte v rozhraní přihlášeného uživatele.

B.2 Přihlášení uživatele

TS2	**Přihlášení uživatele**
Kontext:	Uživatel se úspěšně zaregistroval, ale po poslední návštěvě se odhlásil a musí se znovu přihlásit.
Předpoklady:	Moderátor připraví ověřený účet, pomocí kterého se může uživatel přihlásit.
Úspěch:	Uživatel se úspěšně přihlásil a nachází se v rozhraní přihlášeného uživatele.
Postup:	1. Do prohlížeče zadejte adresu domovské stránky a vyčkejte až se Vám zobrazí domovská stránka s registračním formulářem. 2. Nalezněte odkaz *Login* v horní navigační liště a klikněte na tento odkaz. 3. Vyplňte přihlašovací formulář s poskytnutými údaji, které Vám předá moderátor. 4. Po vyplnění formuláře klikněte na tlačítko *Submit*.

B.3 Změna emailové adresy uživatele

TS3	**Registrace uživatele**
Kontext:	Přihlášený uživatel by si chtěl aktualizovat svojí emailovou adresu.
Předpoklady:	Moderátor připraví novou nezaregistrovanou emailovou adresu, pomocí které si může uživatel aktualizovat současnou emailovou adresu.
Úspěch:	Uživatel si úspěšně aktualizoval emailovou adresu, ověřil novou emailovou adresu a nachází se v rozhraní přihlášeného uživatele.
Postup:	1. V postranní vysouvací liště nalezněte odkaz *Profile* a klikněte na tento odkaz. 2. V obsahu stránky nalezněte tlačítko *Change email* a klikněte na toto tlačítko. 3. Vyplňte formulář pro změnu emailové adresy s poskytnutými údaji, které Vám předá moderátor. 4. Po vyplnění formuláře klikněte na tlačítko *Submit*. 5. Přihlašte se do zadané emailové adresy a ověřte novou emailovou adresu pomocí odkazu ve verifikačním emailu. 6. Po stisknutí na verifikační odkaz se nacházíte v rozhraní přihlášeného uživatele.

B.4 Vytvoření aktivity

TS4	**Vytvoření aktivity**
Kontext:	Přihlášený uživatel by si chtěl definovat novou aktivitu.
Předpoklady:	Moderátor připraví přihlášený ověřený uživatelský účet a nachází se v rozhraní přihlášeného uživatele.
Úspěch:	Uživatel si úspěšně vytvořil novou aktivitu.
Postup:	1. V postranní vysouvací liště nalezněte odkaz *Activities* a klikněte na tento odkaz. 2. V obsahu stránky nalezněte tlačítko *New* a klikněte na toto tlačítko. 3. Vyplňte formulář pro vytvoření aktivity s poskytnutými údaji, které Vám předá moderátor. 4. Po vyplnění formuláře klikněte na tlačítko *Submit*.

B.5 Editace existující aktivity

TS5	**Editace existující aktivity**
Kontext:	Přihlášený uživatel by si chtěl aktualizovat datový interval nebo dny v týdnu aktivity.
Předpoklady:	Moderátor připraví přihlášený ověřený uživatelský účet a nachází se v rozhraní přihlášeného uživatele.
Úspěch:	Uživateli se podařilo aktualizovat datový interval nebo dny v týdnu aktivity.
Postup:	1. V postranní vysouvací liště nalezněte odkaz *Activities* a klikněte na tento odkaz. 2. Zeptejte se moderátora, kterou aktivitu máte aktualizovat. 3. V tabulce s existujícími aktivitami nalezněte zadanou aktivitu a v posledním sloupci klikněte na ikonku pro editaci dané aktivity. 4. Vyplňte formulář dané aktivity s poskytnutými údaji, které Vám předá moderátor. 5. Po vyplnění formuláře klikněte na tlačítko *Submit*.

B.6 Změna stavu aktivity

TS6	**Změna stavu aktivity**
Kontext:	Přihlášený uživatel úspěšně dokončil aktivitu a chtěl by si stav aktivity zaznamenat.
Předpoklady:	Moderátor připraví přihlášený ověřený uživatelský účet a nachází se v rozhraní přihlášeného uživatele.
Úspěch:	Uživatel si úspěšně aktualizoval stav aktivity v daný den.
Postup:	1. Zeptejte se moderátora, kterou aktivitu máte aktualizovat. 2. Aktivitu nalezněte pod horní lištou s logem, kde se nachází aktivity pro daný den. 3. Vyplňte formulář dané aktivity s poskytnutými údaji, které Vám předá moderátor. 4. Po vyplnění formuláře klikněte na tlačítko *Submit*.

B.7 Změna stavu aktivity v minulosti

TS7	**Změna stavu aktivity v minulosti**
Kontext:	Přihlášený uživatel si uvědomil, že v minulosti špatně zadal údaje u aktivity a chtěl by aktualizovat danou aktivitu.
Předpoklady:	Moderátor připraví přihlášený ověřený uživatelský účet a nachází se v rozhraní přihlášeného uživatele.
Úspěch:	Uživatel si úspěšně aktualizoval stav aktivity v minulosti.
Postup:	1. V postranní vysouvací liště nalezněte odkaz *Home* a klikněte na tento odkaz. 2. Zeptejte se moderátora, kterou aktivitu máte aktualizovat. 3. Pokud se zadaná aktivita nenachází v daném týdnu, můžete pomocí šipek nad tabulkou cestovat v časových intervalech, popř. překliknout na měsíční výpis pomocí tlačítka *Monthly* nad tabulkou. 4. Po nalezení aktivity klikněte na buňku tabulky s danou aktivitou. 5. Vyplňte formulář dané aktivity s poskytnutými údaji, které Vám předá moderátor. 6. Po vyplnění formuláře klikněte na tlačítko *Submit*.

B.8 Odhlášení uživatele

TS8	**Odhlášení uživatele**
Kontext:	Uživatel dokončil všechny aktivity a aktualizoval jejich stavy pro daný den a chtěl by se odhlásit.
Předpoklady:	Moderátor připraví přihlášený ověřený uživatelský účet a nachází se v rozhraní přihlášeného uživatele.
Úspěch:	Uživatel se úspěšně odhlásit a nachází se v domovské stránce.
Postup:	1. V postranní vysouvací liště nalezněte odkaz *Logout* 2. Klikněte na odkaz pro odhlášení.

Bibliografie

1. CLEAR, James. *Atomic habits: An easy & proven way to build good habits & break bad ones: Tiny changes, remarkable results* [kniha]. New York, NY: Avery, an imprint of Penguin Random House, 2018 [cit. 2022-03-17]. ISBN 0735211299.
2. *How to start new habits that actually stick* [online]. 2018 [cit. 2022-03-17]. Dostupné z: `https://jamesclear.com/three-steps-habit-change`.
3. *The ultimate habit tracker guide: Why and how to Track yourhabits* [online]. 2019 [cit. 2022-03-18]. Dostupné z: `https://jamesclear.com/habit-tracker`.
4. FITZ-WALTER, Dr Zachary. *What is gamification? education, Business and Marketing (2021 examples)* [online]. Gamify, [b.r.] [cit. 2022-03-19]. Dostupné z: `https://www.gamify.com/what-is-gamification`.
5. KNASTER, Richard. *Advanced topic - domain modeling* [online]. 2021 [cit. 2022-03-17]. Dostupné z: `https://www.scaledagileframework.com/domain-modeling/`.
6. *Habitica* [online]. [B.r.] [cit. 2022-03-13]. Dostupné z: `https://habitica.com/static/home`.
7. *Habitify - The minimal, data-driven habit tracker* [online]. [B.r.] [cit. 2022-03-13]. Dostupné z: `https://www.habitify.me/`.
8. *Coach.me* [online]. [B.r.] [cit. 2022-03-14]. Dostupné z: `https://www.coach.me/`.
9. TKACHENKO, Igor. *Functional vs non-functional requirements: List and examples of systems engineering best practices* [online]. The App Solutions, 2019 [cit. 2022-03-11]. Dostupné z: `https://theappsolutions.com/blog/development/functional-vs-non-functional-requirements/`.
10. MARTIN, Matthew. *What is non-functional requirement in software engineering? types and examples* [online]. 2022 [cit. 2022-03-20]. Dostupné z: `https://www.guru99.com/non-functional-requirement-type-example.html`.
11. KULAK, Daryl; GUINEY, Eamonn. Use cases: Requirements in context, Second edition. In: [kniha]. Addison-Wesley Professional, 2003, s. 29–29 [cit. 2022-03-21]. ISBN 0321154983.
12. ARDALIS. *Overview of ASP.NET core MVC* [online]. [B.r.] [cit. 2022-03-24]. Dostupné z: `https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?WT.mc_id=dotnet-35129-website&amp;view=aspnetcore-6.0`.
13. *Django MVT Architecture* [online]. 2020 [cit. 2022-03-24]. Dostupné z: `https://www.askpython.com/django/django-mvt-architecture`.
14. *Get started* [online]. [B.r.] [cit. 2022-03-25]. Dostupné z: `https://www.djangoproject.com/start/`.

15. *Django databases* [online]. [B.r.] [cit. 2022-03-25]. Dostupné z: https://docs.djangoproject.com/en/4.0/ref/databases/.
16. *Postgresql* [online]. 2022 [cit. 2022-03-25]. Dostupné z: https://www.postgresql.org/docs/current/intro-whatis.html.
17. *Django extensions* [online]. [B.r.] [cit. 2022-03-27]. Dostupné z: https://django-extensions.readthedocs.io/en/latest/graph_models.html.
18. *Graphviz* [online]. [B.r.] [cit. 2022-03-27]. Dostupné z: http://www.graphviz.org/download/.
19. *HTML: Hypertext markup language* [online]. [B.r.] [cit. 2022-04-02]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTML.
20. *CSS* [online]. [B.r.] [cit. 2022-04-02]. Dostupné z: https://www.w3.org/standards/webdesign/htmlcss.
21. *About JavaScript - JavaScript: MDN* [online]. [B.r.] [cit. 2022-04-04]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript.
22. *Ajax - developer guides: MDN* [online]. [B.r.] [cit. 2022-04-05]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX.
23. FOUNDATION, JS. *jQuery* [online]. [B.r.] [cit. 2022-04-05]. Dostupné z: https://jquery.com/.
24. *Chart.js* [online]. [B.r.] [cit. 2022-04-07]. Dostupné z: https://www.chartjs.org/.
25. MARK OTTO, Jacob Thornton. *Bootstrap* [online]. [B.r.] [cit. 2022-04-07]. Dostupné z: https://getbootstrap.com/.
26. *Font awesome* [online]. [B.r.] [cit. 2022-04-08]. Dostupné z: https://fontawesome.com/.
27. *Testing in Django* [online]. [B.r.] [cit. 2022-04-10]. Dostupné z: https://docs.djangoproject.com/en/4.0/topics/testing/.
28. *Django Cache* [online]. [B.r.] [cit. 2022-04-11]. Dostupné z: https://docs.djangoproject.com/en/4.0/topics/cache/.

Obsah přiloženého média

- `readme.txt` ..... stručný popis obsahu média
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF