Ponzi Bank

Technická zpráva k projektu do předmětu ITU
FIT VUT v Brně, 2020

Název týmu
Tým printf

Autoři
David Černý, xcerny74
Lukáš Baštýř, xbasty00
Sotirios Pupakis, xpupak01

Poznámky/pokyny:

- Může se stát, že otázky či doporučení v jednotlivých kapitolách nebudou sedět pro některá zadání. Upravte si tak, aby kapitola obsahovala, co má (třeba i vysvětlení, proč v dané kapitole není co psát).
- Sdělení ve společných částech TZ se chápe jako vyjádření všech členů týmu. Proto je nutno se na něm v týmu shodnout.
- Dodržujte připravené formátování a strukturování textu (pokud logika věci nevyžaduje jinak).
- Obrázky musí obsahovat popisek a musíte se na ně odkázat z textu.
- Pokud není explicitně uvedeno, můžete psát text společně a text je kolektivním dílem.
- Ačkoli jsou reference až na konci dokumentu, odkazy na ně doplňte zejména v první fázi řešení projektu, kdy provádíte průzkum a studium.

Obsah

1. Zadání a organizace týmu

1.1 Cíl

Cílem projektu je vytvořit systém pro transakce virtuálních tokenů které mohou sloužit jako forma měny. Výsledný systém bude obsahovat dvě klientské bankovní aplikace (aplikace pro správu zůstatků na účtech), API se kterou dané aplikace komunikují a rozhraní pro správu celého systému. Systém bude umožňovat transakce bez použití skutečné měny - např. platby mezi hráči nějaké hry nebo pro správu kapesného.

1.2 Tým

Tým se skládá ze tří studentů FITu.

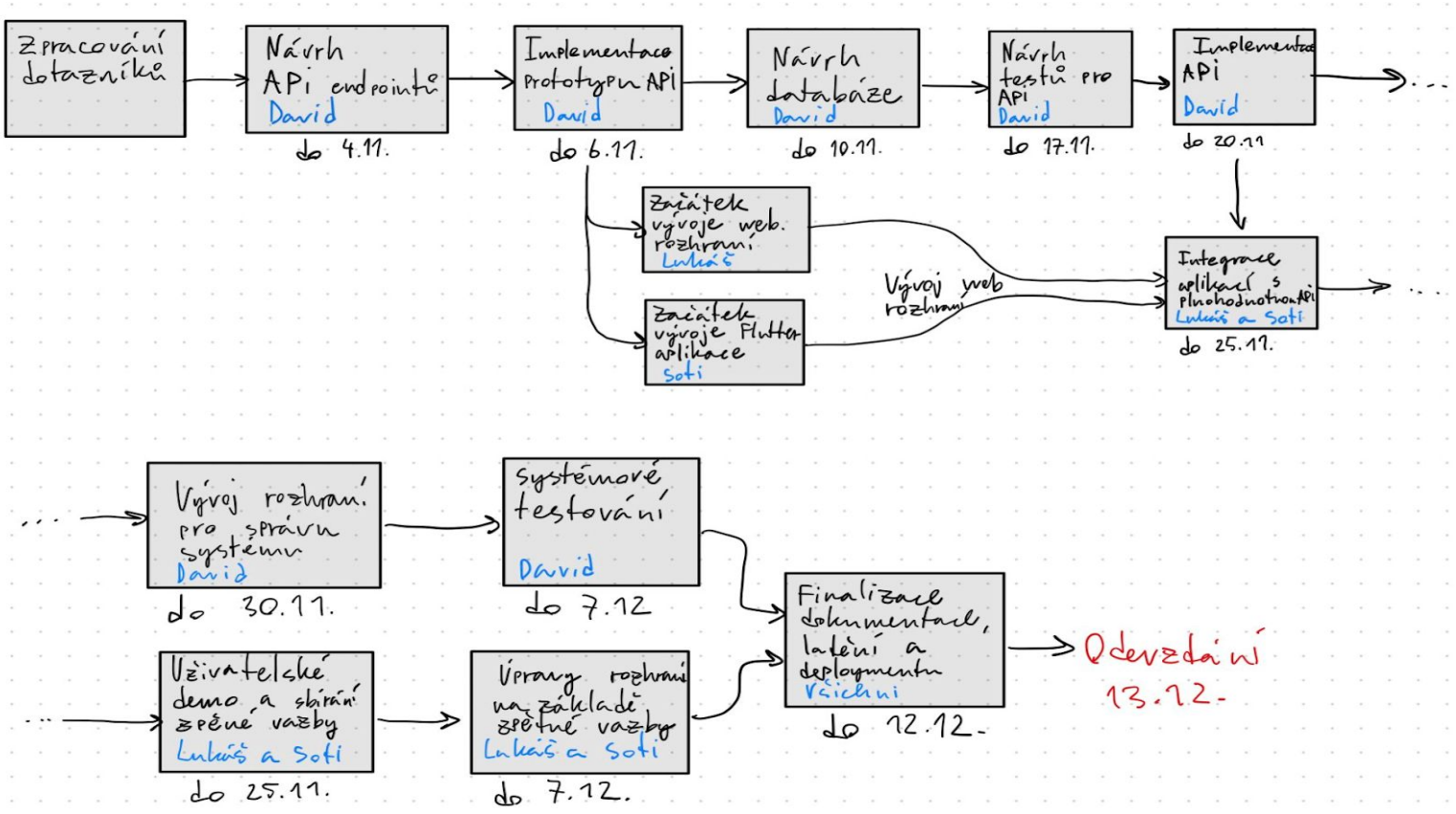
Složení:	David Černý	- David
	Lukáš Baštýř	- Lukáš
	Sotirios Pupakis	- Soti

David má hlavně na starosti API, návrh databáze a webového rozhraní pro správu systému.

Lukáš se podívá na webovou aplikaci pro uživatele, kde budou mít možnost kontroly účtu, posílání peněz a dalších bankovních operací.

Sotiho práce se skládá hlavně z tvorby mobilní aplikaci, která bude mít podobnou funkcionalitu jako webová aplikace pro uživatele.

1.3 Roadmapa



David

1. Zpracování dotazníků
2. Návrh API endpointů (do 4.11.)
3. Implementace prototypu API (do 6.11.)
4. Návrh databáze (10.11.)
5. Návrh testů pro API (10.11.)
6. Implementace API (20.11.)
7. Vývoj rozhraní pro správu systému (30.11.)
8. Systémové testování (7.12.)

Lukáš a Soti

1. Zařátek vývoje webového rozhraní a Flutter aplikace (po implementaci prototypu API)
2. Integrace aplikací s plnohodnotnou API (do 25.11.)
3. Uživatelské demo a sbírání zpětné vazby (do 25.11.)
4. Úpravy rozhraní na základě zpětné vazby (do 7.12.)

Společné

1. Finalizace dokumentace, deploymentu a ladění

1.4 Rizika a opatření

1. Problém se setupem API a databáze na zařízení vývojáře (pro účely vývoje rozhraní)
    - **Řešení 1:** Dockerizace API a databáze, jednotný způsob spuštění skrze docker-compose
    - **Řešení 2:** Spuštění API na veřejné IP adrese pro účely vývoje
2. Nejasnosti se zadáním, neplnění závazků, atd.
    - **Řešení 1:** Sync se všemi členy týmu 1x týdně
        - Každý člen shrne jakou práci vykonal a jaké má plány na další týden
3. Nedokončený/nefunkční backend blokující vývoj
    - **Řešení 1:** vývojáři klientských aplikací dostanou přístup k maketě API která bude poskytovat dummy data aby mohli vyvíjet frontend nezávisle na stavu backendu

2. Průzkum a zkušenosti

2.1 Existující řešení

Inspiraci, jak vizuální tak funkční, jsme přebrali z několika bankovních aplikací pro banky (RB, KB, ČSOB).
Přínosem je lehký přístup k informacím o účtu. U mobilní aplikace je často problém “horšího” zabezpečení. Z aplikace, která je často zabezpečena pinem, je možné přímo posílat peníze. Na druhou stranu ale člověk musí mít přístup k telefonu.

Lukáš

RB webová aplikace

Webová aplikace pro RB[1] dostala v posledních letech vzhledový upgrade, který vedl k lepší přehlednosti základních informací o účtech. Na druhou stranu je nyní značně těžší pracovat s nastavením. Co se mobilní aplikace týče, ta je dělaná hlavně na rychlé QR platby a zjištění stavu účtu (ne/proběhlá platba, zůstatek, poslední transakce). Zabezpečení na webu je pomocí nové aplikace (něco jako google authenticator), která je vázána přímo na účet. Do mobilní je poté potřeba zadat pin nebo pomocí biometrie.

Soti

KB aplikace[2]

Prozkoumal jsem funkčnost mobilní banky od Komerční Banky. Zabezpečení je řešeno buď heslem, KB klíčem, nebo otiskem prstu. Tímto dojde k zjednodušení přihlášení, což je dobrý krok pro uživatele. Dále jsou v aplikaci jednoduše viditelné všechny důležité údaje o účtu a to hned na první obrazovce.

---

[1] "Raiffeisenbank: Banka inspirovaná klienty." https://www.rb.cz/. Datum přístupu: 1 lis. 2020.
[2] "Mobilní banka | Komerční banka." 28 čvc. 2016, https://www.kb.cz/cs/mobilni-banka. Datum přístupu: 1 lis. 2020.

David

ČSOB bankovní aplikace[3]

ČSOB mobilní bankovní aplikace se mi líbí svým čistým designem, má ale i své nedostatky - ne vždy je jasné kde najít klíčová nastavení (např. limity pro platby online), také je obtížné vytváření šablon pro platby.

Ledger Live[4]

Ledger Live není vyloženě bankovní aplikace ale aplikace pro správu kryptoměny, je velmi přehledná ale narozdíl od běžných bankovních aplikací jí zase chybí některé užitečné funkce - vzory pro platby, možnost poslat zprávu příjemci.

2.2 Uživatelské potřeby

Uživatelé budou hlavně lidé, kteří potřebují mezi sebou spravovat nějakou virtuální měnu.

2.3 Shrnutí

Z průzkumu vyšlo, že většině dotázaných stačí praktické informace o účtu (zůstatek a poslední platby) a možnost posílání peněz/tokenů. Co se mobilní aplikace týče, tam by uvítali QR scanner na rychlé platby.

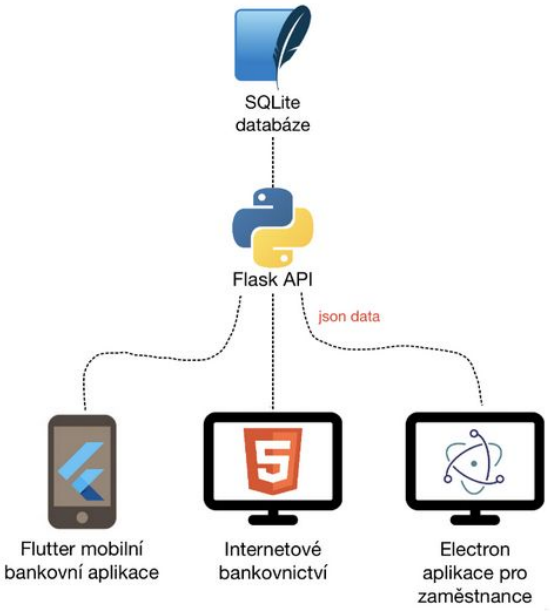
3. Architektura řešení

3.1 Architektura systému

Systém se skládá ze tří částí:

Backend

- **Flask API (Controller)**
    - Zpracovává požadavky zaslané z frontendu a poskytuje data zobrazovaná v aplikacích, pro tyto akce využívá databázi



[3] "Smartbanking, mobilní aplikace na online bankovnictví | ČSOB." https://www.csob.cz/portal/lide/ucty/internetove-a-mobilni-bankovnictvi/smartbanking. Datum přístupu: 1 lis. 2020.
[4] "Ledger Live : Most trusted & secure crypto ...." https://www.ledger.com/ledger-live. Datum přístupu: 1 lis. 2020.

    - Na úspěšné požadavky o přihlášení odpovídá vrácením JWT tokenu[5], který budou aplikace následně používat pro ověření API requestů
    - Veškerá komunikace mezi frontendem a API probíhá ve formátu *json*
- **SQLite databáze (Model)**
    - Ukládá všechna data v systému (informace o účtech, transakcích a zůstatcích)

Uživatelský frontend (View)

- Zobrazuje informace k jednotlivým účtům (zůstatky, historii transakcí, atd.) a umožňuje uživateli efektivně interagovat s API

Administrátorský frontend (View)

- Zobrazuje informace o provozu systému (seznam uživatelů, seznamy transakcí, informace o databázi a API, logy, atd.)

3.2 Architektura aplikace/í

Lukáš

Webová aplikace (React)

Pro webovou aplikace hodlám použít React (JS knihovna pro vývoj UI), díky které by měla aplikace fungovat na moderních/běžných prohlížečích. Vzhledem k tomu, že se jedná o webovou aplikaci, bude potřeba také použít HTML.
Samotná aplikace bude frontend (view) pro naše společné API (backend - controller). Jako model bude použita databáze sqlite.

Soti

Flutter mobilní aplikace

Mobilní aplikace bude psána ve frameworku Flutter, který zajistí stejné zobrazení napříč všemi bolními zařízeními. Flutter jako takový je stavěn na MVC architektuře, kterou budu tedy používat.

David

Rozhraní pro správu systému

Frontend aplikace slouží jako *View* zobrazující data získaná z API. Backend aplikace bude mít na starosti komunikaci s API - zpracování příchozích dat a vygenerování odchozích requestů. API slouží jako *Controller* (validuje správnost příchozích dat a komunikuje s modelem) SQLite databáze slouží jako model obsahující všechna data

3.3 Datový model

- **Data o klientovi**

[5] "JWT.io." https://jwt.io/. Datum přístupu: 1 lis. 2020.

    - Uživatelské jméno
    - Číslo/identifikátor účtu
    - Zůstatky pro každou měnu
    - Hash hesla
    - Typ účtu (administrátor/běžný uživatel)
    - Nastavení účtu (heslo, atd.)
    - Je účet zablokovaný?
- **Transakce**
    - Čas transakce
    - Měna transakce
    - Objem transakce
    - Zpráva příjemci
    - Odesílatel
    - Příjemce
    - Je transakce “anonymní”? (pokud to daná měna povoluje)
- **Nastavení systému**
    - Jak dlouho je platný JWT token?
    - Měny a jejich nastavení

API endpointy

- **/api/v1/** - verze api (v budoucnu může dojít k úpravám)

Veřejné endpointy

- **[POST] /api/v1/auth** - zaslání přihlašovacích údajů
    - V případě platných údajů vrátí endpoint JWT token
    - Odhlášení není třeba oznamovat serveru, stačí zahodit JWT token a pro opětovné přihlášení požádat server o nový

Zabezpečené endpointy

Při volání API není třeba určovat identitu uživatele, ta je získána z JWT tokenu.

Klientské

- **[GET] /api/v1/user_data** - získá základní data o uživateli (pouze pro aktuálně přihlášeného uživatele)
- **[GET] /api/v1/settings** - získá aktuální nastavení uživatele
- **[POST] /api/v1/settings** - změní aktuální nastavení uživatele (atributy které chceme změnit jsou v tělerequestu, ostatní zůstanou nezměněné)
- **[GET] /api/v1/balances** - získá zůstatky pro každou měnu
- **[GET] /api/v1/currencies** - získá seznam všech měn a jejich nastavení
- **[GET] /api/v1/<currency>/transactions** - získá historii transakcí pro danou měnu
- **[POST] /api/v1/send** - pošle transakci (informace o transakci jsou v těle requestu)

Administrátorské

- **[GET] /api/v1/users** - získá základní informace o všech uživatelích
- **[GET] /api/v1/users/<user_id>** - získá detailnější informace o uživateli

- **[POST] /api/v1/users/<user_id>** - změní nastavení uživatelského účtu (blokace, atd.)
- **[DELETE] /api/v1/users/<user_id>** - smaže uživatelský účet
- **[GET] /api/v1/transactions** - získá log transakcí (posledních n transakcí)
- **[GET] /api/v1/export/<target>** - exportuje požadovanou komponentu (logy, databázi)
- **[GET] /api/v1/import** - importuje databázi (nahradí stávající databázi)
- **[POST] /api/v1/<currency>/settings** - edituje nastavení měny (atributy které chceme změnit jsou v těle requestu, ostatní zůstanou nezměněné)
- **[POST] /api/v1/restart** - restartuje API a databázi

3.4 Vybrané technologie a implementace

Pro backend jsme se rozhodli použít *Flask* knihovnu a databázi *SQLite*. Pro autentizaci uživatelů budou použity *JWT (JSON Web Tokens)*. Pro rozhraní správy systému bude použit framework *Electron*. Pro webové rozhraní pro uživatele bude použit framework *React*.
Pro mobilní aplikaci bude použit framework *Flutter*.

4. Návrh GUI

4.1 Webová aplikace [xbasty00]

4.1.1 Požadavky na GUI

Možnost zobrazení informací o účtu, různá vyplňovací okna pro zadávání hodnot. Možnost upravit některá nastavení (například změna jazyka) - dropdown menu.

4.1.2 Makety

Interaktivní wireframe rozhraní, PDF verze wireframu

4.1.3 Pilotní test

Testy proběhly na běžných uživatelích pomocí zaslání makety. Následně jsem je požádal, jestli jsou schopni poslat na cizí účet prostředky ze svého účtu a následně si zobrazit historii pohybů na účtu. Vzhledem ke stavu jsem doufal na

4.1.4 Vyhodnocení testu a revize návrhu

Nikdo údajně neměl problém se splněním těchto zásadních, ale také nejdůležitějších úkolů. Jediné, co se uživatelům nelíbilo bylo nezarovnání horních ikon s hlavním menu barem. Jinak se uživatelům UI líbilo a neměli s ním problém.

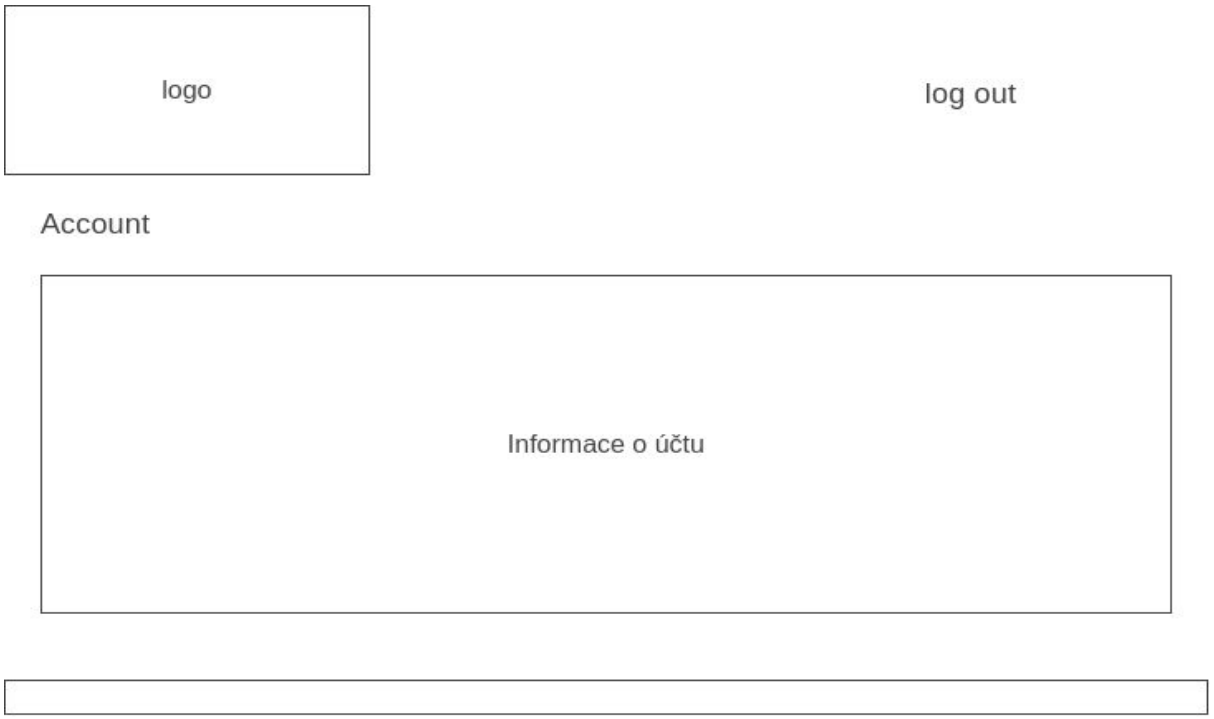
4.2 Mobilní aplikace [xpupak01]

4.2.1 Požadavky na GUI

GUI bude hlavně přehledně zobrazovat uživateli všechny informace o účtu, jako zůstatek, provedené transakce, číslo a další. Také bude umožňovat uživateli pohodlně zadat odchozí platbu.

4.2.2 Makety


lišta
informace o účtu + rychlé akce
Historie
Transakce
Tansakce
Transakce
Transakce
Transakce
Transakce



4.2.3 Pilotní test

Pomocí wireframů jsem na své rodině otestoval, jak se jim líbí rozložení a intuitivnost aplikace. Hlavně jestli dosáhnout na nejdůležitější prvky jednou rukou a nemusí si mobil přidržovat.

4.2.4 Vyhodnocení testu a revize návrhu

Všichni byli s návrhem spojeni a neměli žádné závažné připomínky. Dostal jsem pouze doporučení na barvy.

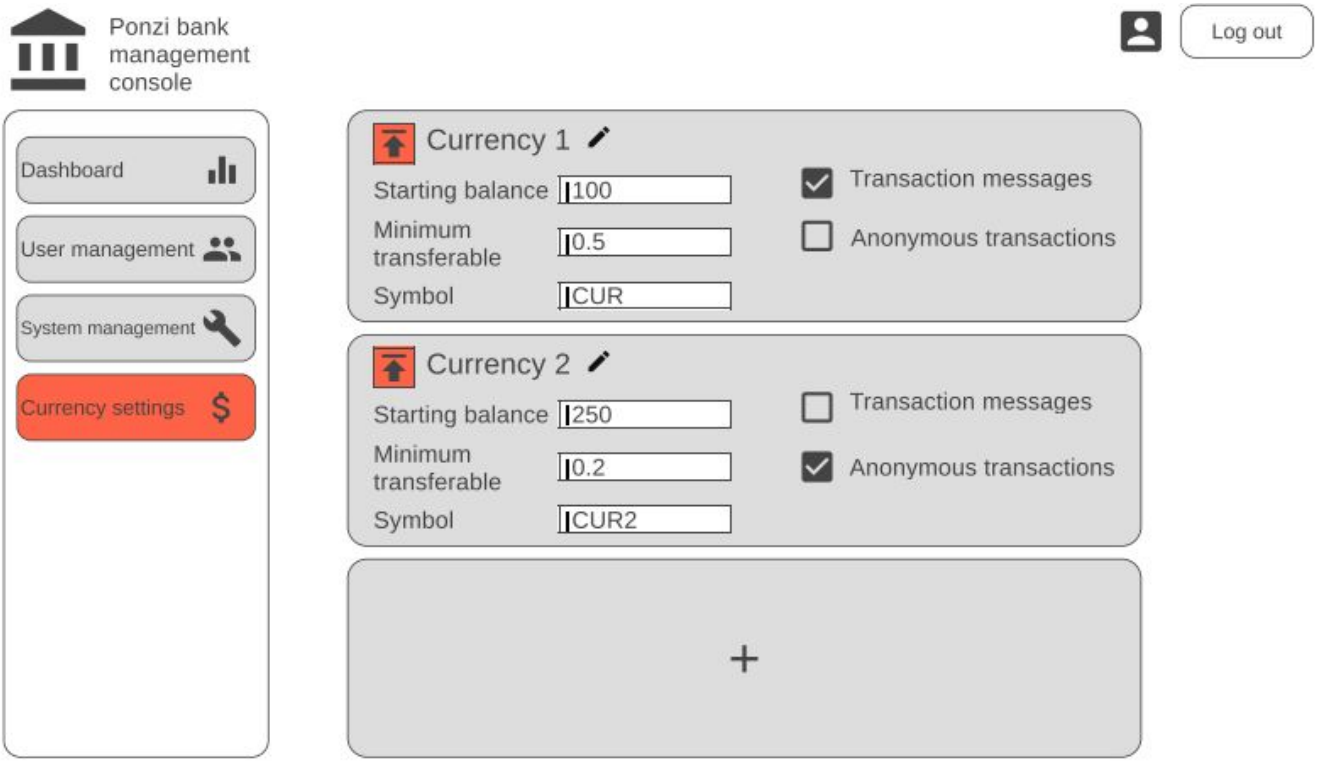
4.3 Rozhraní pro správu systému [xcerny74]

4.3.1 Požadavky na GUI

Administrátor by pomocí rozhraní pro správu systému měl mít možnost získat rychlý přehled o stavu systému: aktuálně vypisované logy, aktuální vytížení systému, atd. Dále by měl pomocí rozhraní být schopen rychle spravovat uživatele (přidávat, odstraňovat, upravovat uživatelské účty). Důležitá je i správa jednotlivých měn - v rozhraní bude možné přidávat různé virtuální měny/tokeny a nastavovat jejich chování - výchozí zůstatek dané měny při vytvoření účtu, nejmenší přenositelná částka, zda lze posílat transakce v rámci systému anonymně, zda je možné posílat zprávu příjemci, atd. Také bude v rozhraní možné exportovat a importovat databázi, celý systém restartovat a exportoval logy o provozu systému.

4.3.2 Makety

[Interaktivní wireframe rozhraní](), [PDF verze wireframu]()



4.3.3 Pilotní test

Pilotní test byl realizován na uživateli středního věku s mírně nadprůměrnými uživatelskými schopnostmi (používá matlab, je schopen nainstalovat OS), není to ale IT profesionál. Uživateli bez předchozích znalostí s rozhraním (viz. interaktivní wireframe) byly podány následující úkoly:

- **Přihlásit se do systému** - proběhlo bez problému
- **Přidat novou měnu** - taktéž proběhlo bez problému
- **Přidat nového uživatele** - uživatel rychle rozpoznal příslušné tlačítko, vše bez problému
- **Restartovat systém** - uživatel se místo restartování systému odhlásil (to bylo možná ale způsobeno špatnou formulací požadavku), na druhý pokus už úkol splnil bez problému

Uživateli se GUI celkově líbilo, nic by na něm neměnil.

4.3.4 Vyhodnocení testu a revize návrhu

Výsledky testu ukázaly, že rozhraní je dostatečně intuitivní na to, aby i uživatel bez zkušeností byl schopen vykonat zadané úkony - napomáhají tomu výrazná tlačítka doprovázená ikonami. V návrhu není potřeba v podstatě nic měnit, bude ale ještě zhodnoceno zda tlačítko pro restart systému neumístit přímo do hlavního okna aplikace.

Druhá část

---

5. Implementace GUI - aplikace XY (nebo část aplikace XY) [login autora]

Tato kapitola včetně podkapitol musí být vypracovaná každým členem týmu zvlášť, na jeho vlastní část GUI aplikace, kterou bude autorsky řešit.
Kapitola bude mít číslo pravděpodobně vyšší, podle počtu kapitol s Návrhem GUI (tj. podle počtu členů týmu).

5.1 Implementace

Popište důležité části implementace GUI, propojení GUI s funkcemi a datovým modelem, popište případně pomocné datové struktury atd.

5.2 Použité nástroje a knihovny

Popište jaké knihovny či jejich části a funkce byly využity při implementaci GUI a jakým způsobem. Shrňte vlastní zkušenost s těmito nástroji a technologiemi (výhody, potíže atd.).

5.3 Finální testování

Stejný postup jako u Pilotního testu výše.

5.4 Vyhodnocení testu

Stejně jako u vyhodnocení Pilotního testu. Výsledky diskutovat, není nutná revize návrhu, stačí popis, co a jak je nutné do budoucna udělat jinak.

6. Závěr

Stručně popište dosažený výsledek a hlavní zjištění z testování. Důležité uživatelské zkušenosti nepopisujte obecně (“uživatelům se to líbilo”), ale co nejvíce konkrétně (co a jak uživatelé použili, zvládli, pochopili, ocenili).
Zakončete krátkým shrnutím Vaší zkušenosti s prací celého týmu a Vaší roli. Co pro Vás bylo při práci v týmu přínosné (2-3 věci) a co byste příště udělali jinak (pouze 1 věc).
Kapitola bude mít číslo pravděpodobně vyšší, podle počtu kapitol s Návrhem GUI a s Implementací (tj. podle počtu členů týmu).

Reference

Sem pište zejména důležité zdroje, odkud jste čerpali informace (knihy, články, časopisy, blogy apod.).
Odkazy na SW, manuály, fóra apod. uvádějte formou url do poznámky pod čarou.