DELTA – Střední škola informatiky a ekonomie, Základní škola a Mateřská škola s.r.o.
Ke Kamenci 151, PARDUBICE

MATURITNÍ PROJEKT
GLOBÁLNÍ HELPDESK

Skořepa, Viktor
4.B
Informační technologie 18-20-M/01
2021/2022

Zadání maturitního projektu z informatických předmětů

Jméno a příjmení:	Viktor Skořepa
Školní rok:	2021/2022
Třída:	4. B
Obor:	Informační technologie 18-20-M/01
Téma práce:	Globální helpdesk
Vedoucí práce:	Vlaďka Janů

1. Způsob zpracování, cíle práce, pokyny k obsahu a rozsahu práce:

Cílem projektu je vytvořit internetovou platformu na pomoc při potížích, konkrétně na pomoc s počítači lidem, kteří tomu nerozumí a nemají čas si něco hledat a složitě se to učit. Inspirací je platforma Uber.

Aplikace by měla fungovat tak, že lidé, co rozumí počítačům a chtějí si přivydělat se zde zaregistrují jako operátoři, účet operátora musí schválit admin / vlastník aplikace na základě jeho žádosti o přijmutí. Operátor by měl v žádosti důkladně popsat v čem je dobrý, s čím má zkušenosti a podle toho mu bude přiřazen obor problémů, které se mu budou zobrazovat. Např.: Operátor je grafik s dlouhou zkušeností s programy od Adobe, budou se mu zobrazovat problémy s programy Adobe, ale třeba i pomoc obnovy hesla u Adobe apod.

Druhá část aplikace je právě pro lidi, co počítačům nerozumí a potřebují s něčím poradit. Ty se zde zaregistrují a po ověření e-mailu už mohou zakládat svůj problém. Poté jim bude přidělen operátor, který má specializaci pro dané kategorie, které si zákazník vybral. Ten se pokusí problém vyřešit. Pokud toho nebude schopen, může z problému odstoupit, ticket se znovu otevře a může si ho přiřadit jiný operátor, který už řešení třeba bude vědět.

Aplikace negarantuje vyřešení problému!! Všechno záleží na komunitě.

1. Část vidí zákazník, tedy člověk, který má nějaký problém na počítači a přišel ho sem řešit. Část by měla obsahovat hlavně možnost vytvoření nějakého ticketu, poté profil uživatele, změnu hesla, možnost zobrazit si svoje předchozí tickety.
2. Část vidí operátor stránky, který se zákazníky řeší jejich problémy. Vidí všechny nevyřízené tickety, ať už pouze v jemu přidělených kategoriích nebo všechny. Tickety může zavírat jako vyřešené, může ale z ticketu i odstoupit a znovu ho otevřít např. z důvodu, že neumí takovýto problém řešit.
3. Část vidí admin stránky, který vidí historii všech ticketů, může je úplně smazat nebo znovu otevřít. Hlavní funkcí je správa operátorů. Může přijímat jejich žádosti, ale také blokovat operátory, kteří se prokázali jako velmi neúspěšní a nejsou schopni řešit žádné problémy.

Tickety by měly být řešeny způsobem okamžité pomoci. Součástí založení ticketu je nějaký předmět, stručný popis problému, a hlavně výběr kategorií problému. Kategorie jsou předpřipravené. Zákazník si také může zvolit závažnost problému, přičemž závažnějším problémům budou dávat operátoři přednost, a to za větší minutovou sazbu. Také musí mít čas jeho založení a ukončení, jméno operátora, který ticket řešil a po vyřešení také zhodnocení přístupu operátora.

2. Stručný časový harmonogram (s daty a konkretizovanými úkoly):

- 1. - 7. září 2021: výběr technologií, které budou na projektu použity
- 7.září. - 30. září 2021: začátek projektu – návrh layoutu + začátek implementace vytváření ticketů
- 1.října – 30. října 2021: dodělání systému ticketů, ticket by měl jít vytvořit, řešit a zavřít
- 1. listopadu – 31. prosince 2021: implementace přihlašování se 3 různými právy, registrace zákazníku a registrace operátorů
- 1. – 30. leden 2022: dodělávky detailů, jako je třeba hodnocení operátorů, psaní dokumentace
- 1. února – 31. března 2022: ladění funkčností, testování, spuštění produkční verze

Prohlašuji, že jsem maturitní projekt vypracoval(a) samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 31.3.2022

Anotace

Práce dokumentuje postup vývoje webové aplikace na pomoc uživatelů nezkušených s IT s jejich základními problémy s počítačem pomocí chatu. Aplikace po spuštění bude schopna žít si vlastním životem bez nutnosti správy. Je napsána ve frameworku Next JS s použitím open source BaaS Supabase.

Anotation

This document documents the development process of a web application with the goal of helping inexperienced computer users with their struggles with computers and technology through a real time chat with an operator. Application should be able to function on its own, without the need of any outside maintenance. It is written in the Next JS framework and an open source BaaS Supabase.

Obsah

Úvod

Na začátek by bylo dobré ujasnit si, jak nápad na projekt vznikl a co mě k němu vedlo. V dnešní době se stávají decentralizované aplikace (firmy) stále populárnějšími. Nejvíce to odstartovala společnost UBER, která dnes funguje po celém světě a využívají jí statisíce zákazníků i řidičů. Filozofie za těmito aplikacemi je vzít něco, co ve světě funguje dlouho a je to využívané. A udělat to tak jednoduché, jak jenom to jde. Často s využitím moderních technologií. Jakmile napíšete takovou aplikaci dokonale, stačí jen chvíli počkat a začnou se dít věci. Takovéto aplikace ale píší skupiny desítek programátorů několik měsíců. Cílem tedy není napsat již zmíněnou dokonalou aplikaci, ale zkusit si, jak daleko se může maturant za půl roku dostat a tuto část případně využít v budoucnosti při přepisování nebo dodělávání aplikace. Tyto aplikace již fungují na dovoz jídla, dopravu, nákup potravin a doplňků. Na internetový helpdesk se samozřejmě také můžete přihlásit pomocí aplikace, ale, abyste byli operátor, musíte se jimi nechat zaměstnat, a to je proces, který jen tak nějaký student vysoké školy podstoupit nechce. Úplný cíl, tedy ne tohoto projektu, je napsat aplikaci tak, aby se během několika minut povedlo uživateli vyřešit jeho problém, ale aby se i operátorovi povedlo vydělat 10 korun a jít řešit problém další.

1. Teoretická část

1.1. Výběr technologií

Při výběru technologií byla snaha zkombinovat jak technologie, se kterými už jsem někdy pracoval, tak i nějaké nové, abych se něco přiučil. Celý projekt je napsán v Next.js, na backendu je použitá supabase, ze které si data na frontend dostávám pomocí ReactQuery. Na CSS styly je použita knihovna tailwind. Projekt využívá pár menších knihoven.

1.1.1. Next.js

Jedna z hlavních technologií na mém projektu je Next JS. Next JS je open-source framework postavený na React.js. Je vyvíjený společností Vercel. Next.js obohacuje React o funkcionality jako jsou renderování na serveru nebo generování statických stránek.[1]

1.1.2. React

React je v dnešní době velmi využívaná frontend knihovna. Pomocí Reactu můžeme jednoduše vytvářet jednotlivé komponenty, ze kterých poté webové stránky skládáme dohromady. React využívá JSX.[2]

1.1.2.1. JSX

JSX je upravená syntaxe podobná HTML využívána reactem, která zjednodušuje psaní Javascriptu a HTML dohromady. Pomocí Babelu se poté vše transformuje do obyčejného JavaScript kódu. Je tomu naopak než dříve, kdy se psal JavaScript kód do HTML, nyní píšeme HTML do JavaScript kódu. [3]

1.1.2.2. Babel

Babel je kompilér, který překompiluje JSX kód do čistého Javascriptu. Využívá ho například React a jak vidíte v ukázce níže, kompiluje z JSX – velmi uživatelsky přívětivého kódu do již trochu zmateného Javascriptu. Velmi tak usnadňuje práci programátorům. [3]

JSX:

kód 1

```
var nav = (
   <ul id="nav">
     <li><a href="#">Home</a></li>
   </ul>
);
```

JavaScript:

Kód 2

```
var nav = React.createElement(
  "ul",
  { id: "nav" },
  React.createElement(
    "li",
    null,
    React.createElement(
      "a",
      { href: "#" },
      "Home"
    )
  )
);
```

[3]

1.1.2.3. Supabase

Jako databáze byla vybrána Supabase. Je to open source alternativa k Firebase. Jedná se o cloudovou službu, která nabízí tzv. backend-as-a-service (BaaS). [4]

Supabase je postavená na PostgreSQL což je moderní open source SQL databáze s mnoha funkcemi.[6] Nabízí navíc od databáze i např. přihlašování, uložiště pro videa či fotky, nebo i vlastní serverové funkce a cron joby. Má velmi přívětivou verzi zdarma až na 2 projekty na uživatele. Není ale problém si připlatit a použít Supabase na větší projekty.

Supabase je jeden z nejrychleji rostoucích open source projektů na světě. Jelikož se komunitě velmi líbí použití BaaS nad velmi stabilní databází jako je PostgreSQL. Supabase je dobrá i v tom, že nám dovoluje přistupovat do databáze skrze jejich API, které mimochodem i dynamicky vytváří API dokumentaci přímo k vašemu projektu, ale dává uživateli i přístup přímo k samotné PostgreSQL databázi v případě, že uživateli bude chybět v tomto směru jakákoliv funkce. Nejen, že má možnost požádat velmi ochotnou komunitu, která reaguje velmi rychle a

při rozumném požadavku by netrvalo dlouho funkci do Supabase doprogramovat. Tak může využít i funkce PostgreSQL napřímo. [5]

1.1.2.3.1. BaaS

Je Cloudová služba, která nabízí zákazníkům předat všechny procesy, co se dějí na pozadí aplikace, do nějakého cloudu, který to za ně vyřeší. Odpadá tak nutnost programátorů soustředit se na backend a místo toho pomocí API využívají služby nějakého BaaS poskytovatele. Dále i odpadá nutnost spravovat si vlastní servery, o to se v cloudu také stará poskytovatel služby. [5]

1.1.2.4. Tailwind

Tailwind je CSS knihovna, která velmi usnadňuje stylování vaší aplikace. Styl jakékoliv vaší komponenty píšete přímo do její třídy, usnadňuje tak stylování komponent. Také optimalizuje posílání stylů do vaší aplikace, nikdy nepošle styly, které nebudou využity. Nakonec i zjednodušuje udělat stránku responzivní.[7]

1.2. Výběr projektu

Při výběru tématu maturitního projektu jsem hledal něco, co by dávalo smysl a co by mě mohlo bavit. Snažil jsem se najít něco, co má potenciál, ale ještě není na trhu. Když se řekne helpdesk, nejspíše si řeknete, že to není nic revolučního. Když se nad tím trochu více zamyslíte dojde vám, že poslední dobou nejsou úspěšné naprosto nové revoluční projekty, ale projekty, které vezmou něco naprosto nerevolučního a předělají to revolučně s využitím moderních technologií.

1.2.1. Inspirace

Pro inspiraci nemusíme chodit vůbec daleko. Snad již každý se dnes na dovolené nebo snad jen při spěchu na nádraží v Praze svezl Uberem. Uber není žádná novinka, je tu s námi již několik let, ale novinka to nebyla ani při jeho vzniku. Takové staré dobré taxi je tu s námi již desítky, možná stovky let, a i přes to se uberu podařilo být v tomto odvětví tak strašně úspěšní. Je to právě Uber, který mě inspiroval začít tento projekt a od začátku si říkám, když tento konceptu může fungovat v prostředí, kde někdo může přijít o život, proč by nemohl fungovat v bezpečném prostředí internetu.

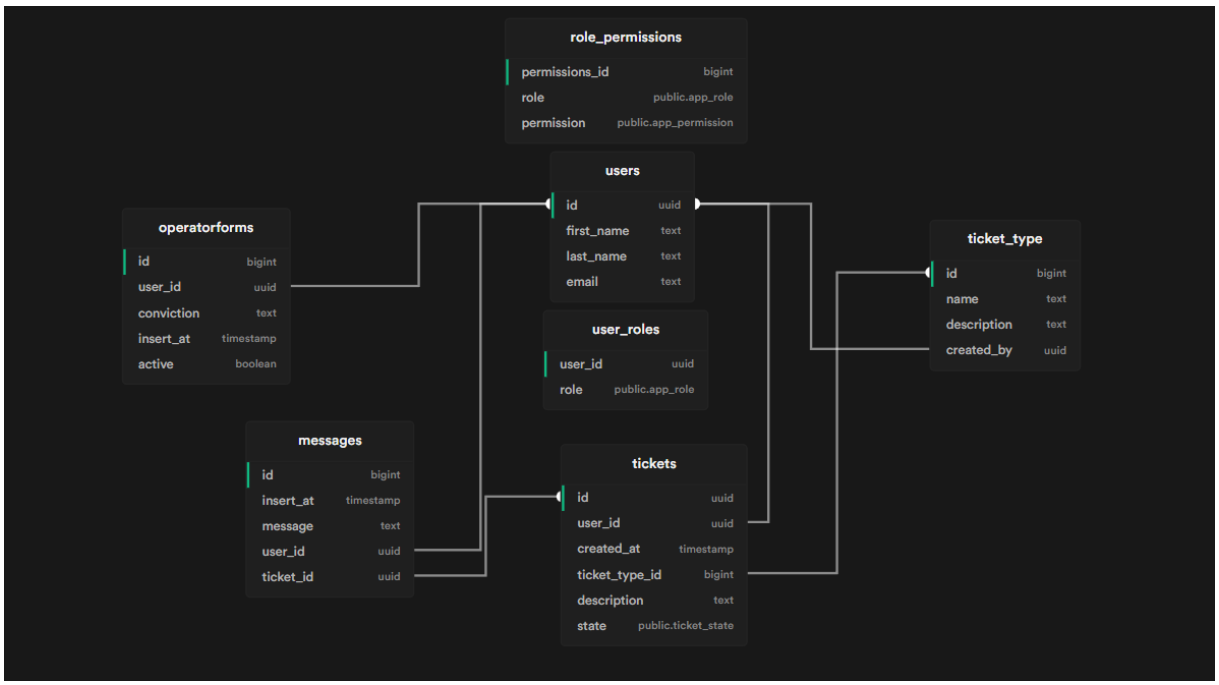
2. Praktická část

2.1. Schéma databáze

Databáze má velmi jednoduché schéma viz obrázek č. 1. Jsou zde základní komponenty pro funkčnost aplikace: tickety, zprávy, uživatelé a typy ticketů. Poté dvě pomocné tabulky user_roles a role_permissions, ve kterých je uložena role uživatele, ty musí být ve zvláštní tabulce, abychom jednoduše mohli zabránit tomu, aby si uživatel mohl roli změnit třeba na

admina. V tabulce role_permissions definujeme oprávnění operátorů a adminů. Tedy např. admin má právo měnit roli uživatelů, ale operátor nikoliv.

Obrázek 1 schéma databáze



2.2. Funkce na databázi

Supabase nabízí uživateli napsat si vlastní funkce v jazyku PLPGSQL, které se budou moci volat na straně databáze. Tyto funkce se využívají např. k řešení přístupů k datům. Nebo se dají propojit s nějaký triggerem, který se spustí stane-li se něco. Díky tomu může být aplikace napsána bez backendu, jelikož funkce na databázi mohou obejít bezpečnostní pravidla nastavené pro veřejný api klíč. A tak se postarat o některé věci, které chceme udělal, ale nechceme povolit z venku.

2.2.1. Funkce authorize

Obrázek 2 funkce authorize

```
create or replace function public.authorize(
  request_permission app_permission,
  user_id uuid
)
returns boolean as $$
declare
  bind_permissions int;
begin
  select count(*)
  from public.role_permissions
  inner join public.user_roles on public.role_permissions.role = user_roles.role
  where public.role_permissions.permission = $1
    and user_roles.user_id = $2
  into bind_permissions;

  return bind_permissions > 0;
end;
$$ language plpgsql security definer;
```

Funkce authorize je PLPGSQL funkce, která je napsaná na databázi. Stará se o to, aby uživatelé, kteří mají mít přístup k některým datům navíc ho mít budou. Má 2 parametry, a to request_permission a user_id. Funkce si sloučí tabulky user_roles a role_permissions podle role. Vznikne tabulka se 3 řádky, kde jsou všechny 3 role a k rolím přiřazené jejich pemissions z tabulky role permissions, kde jsou vytvořené záznamy s rolí a oprávněním, které má. Funkce potom hledá řádek, na kterém je role vybraného uživatele podle parametru user_id a pokud má tato role dané oprávnění podle parametru request_permission viz obrázek č. 2. Pokud ano vrátí funkce true a funkce v RLS povolí uživatele přístup k datům v dané tabulce.[8]

2.2.2. Funkce isinmyticket

RLS nám jednoduše dovoluje dát uživateli přístup ke zprávám, které nesou jeho user_id. Uživatel, ale potřebuje přístup i ke zprávám, které někdo poslal jemu, právě v jeho ticketu. O to se stará právě funkce isinmyticket.

Obrázek 3 funkce isinmyticket

```
create or replace function public.IsInMyTicket(
  v_ticket_id uuid,
  v_user_id uuid
)
returns boolean as $$
declare
  rec record;
  v_tickets int := 0;
begin
  for rec in select id, description from public.tickets where user_id = $2
  loop
    if rec.id = $1 then
      v_tickets := v_tickets + 1;
    end if;
  end loop;

  return v_tickets > 0;
end;
$$ language plpgsql security invoker;
```

Funkce dostane 2 parametry ticket_id a user_id projde všechny tickety daného uživatele, pokud je některý z nich ticket definovaný v parametru, vrátí true viz obrázek č. 3. Pokud se tedy jedná o ticket uživatele, povolíme mu číst tyto zprávy, pokud ne, zprávy nemá jak získat.

2.2.3. Funkce handle_new_user

Obrázek 4 funkce handle_new_user

```
create or replace function public.handle_new_user()
returns trigger as $$
begin
  insert into public.user_roles (user_id, role)
  values (new.id, 'user');
  return new;
end;
$$ language plpgsql security definer;
```

Tato funkce je zavolána při vytvoření nového uživatele. V Supabase je pro uživatele již vytvořená tabulka, kterou si uživatel nemůže moc upravit. Pro ukládání rozšiřujících dat uživatele je vytvořena v projektu další tabulka s uživateli. Jakmile se do nativní tabulky uživatelů vloží nový uživatel, pomocí API funkce signup() zavolá se funkce handle_new_user(), která vytvoří záznam o tomto uživateli v naší tabulce uživatelů viz obrázek č. 4. My si do ní poté můžeme vložit jakékoliv rozšiřující informace, v našem případě jméno a příjmení.

2.2.4. Funkce NoWaitingTicketFunkce

byla vytvořena jako ochrana proti spamu. Prověří, zda uživatel nemá žádný ticket se stavem waiting viz obrázek č. 5. Pokud ano, nepovolí uživateli vložit nový záznam do databáze neboli nový ticket

Obrázek 5 funkce NoWaitingTickets

```
create or replace function public.NoWaitingTickets(
  v_user_id uuid
)
returns boolean as $$
declare
  rec record;
  v_tickets int := 0;
begin
  if auth.uid() != $1 then
    v_tickets := v_tickets + 1;
  end if;
  for rec in select id, state from public.tickets where user_id = $1
  loop
    if rec.state = 'waiting' then
      v_tickets := v_tickets + 1;
    end if;
  end loop;

  return v_tickets <= 0;
end;
$$ language plpgsql security invoker;
```

2.3. Bezpečnost

Bezpečnost je naprostou nutností jakékoliv internetové aplikace. Bez možnosti přihlášení a správy uživatelů nemáte žádnou možnost, jakkoliv moderovat uživatele, kteří se nebudou chovat, jak by měli.

2.3.1. Autentizace

Autentizace je první a důležitou částí bezpečnosti aplikace. Autentizace se stará o to, aby se uživatel mohl přihlásit jen k účtu, který je právě jeho. K tomu samozřejmě patří i registrace. Zjednodušeně zařizuje to, abyste se s vaším emailem a heslem mohli přihlásit.

V mé aplikaci toto řeší supabase. Má na autentizaci předpřipravené API funkce, které jsou v aplikaci používány. Předáte funkci potřebná data, při registraci jich je více, při přihlašování pouze email a heslo. A pokud je přihlášení úspěšné, přidá se do session záznam s JWT tokenem, který obsahuje všechny potřebné informace.

2.3.2. Autorizace

Díky tokenu v session víme, zda je uživatel přihlášen, a co je to za uživatele. Toto je využito k rozlišení, jestli se jedná o obyčejného uživatele, operátora nebo admina. Každý z těchto uživatelů má jiná oprávnění, má přístup na jiné stránky, na některých stránkách jiné uživatelské rozhraní a má přístup k jiným datům z databáze.

2.3.3. Přístup k datům

Díky tomu, že Supabase je postavená na PostgreSQL nabízí nám použití RLS (row level security). Tato služba spravuje přístup uživatelů k datům v tabulce na základě námi definovaných podmínek. Pro jednotlivé tabulky máme několik podmínek.

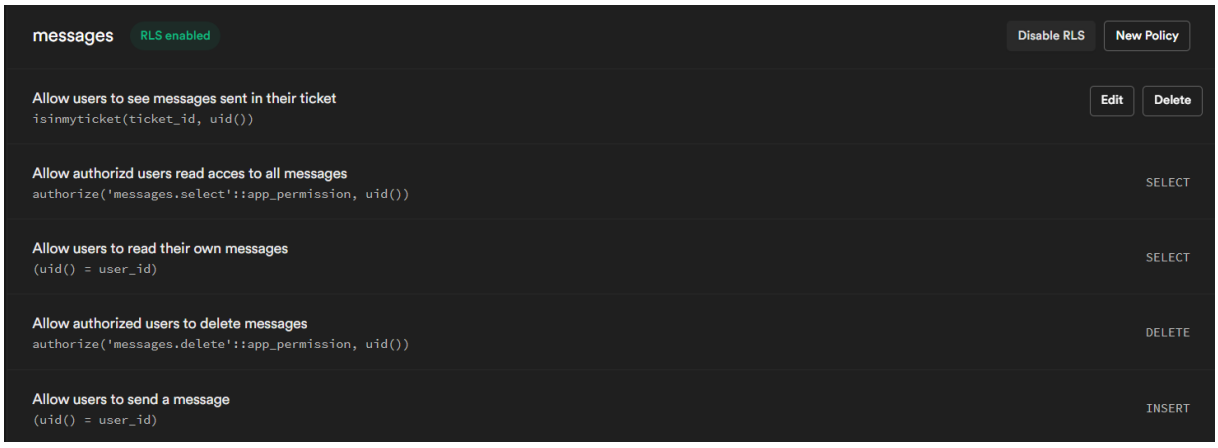
2.3.3.1. Tabulka ticketů

Obrázek 6 RLS – tabulky ticketů

Tabulka ticketů má jednoduchá pravidla pro vytváření, čtení a upravování ticketů s vaším id. A poté jsou zde ještě pravidla pro autorizované uživatele viz obrázek č. 6.

2.3.3.2. Tabulka zpráv

Obrázek 7 RLS – tabulka zpráv



Tabulka se zprávami má několik pravidel viz obrázek č. 7. Uživatel může vložit do tabulky záznam se svým id neboli může poslat zprávu. Uživatel také může číst zprávy, které on poslal, ale i ty, které poslal někdo jemu.

2.4. Funkce Aplikace

Aplikace má několik funkcí, od registrace až po řešení ticketů pomocí real time konverzace.

2.4.1. Registrace

Pro registraci aplikace využívá předpřipravené funkce od supabase. Jako hlavní auth provider využívá email a heslo. Uživatel zadá do formuláře jméno, příjmení, email a heslo a na email mu přijde potvrzovací odkaz, po jehož prokliknutí se jeho účet aktivuje a může se přihlásit. Formulář je napsán pomocí react hook forms. Má tedy kontrolu na straně klienta, a to jak pomocí html, tak javascriptu. Formulář se poprvé načte při načtení stránky, během vyplňování se neobnovuje a tím pádem ani nekontroluje svoje podmínky. Ty zkontroluje až po pokusu o odeslání. Pokud nějakou podmínku nesplňujete, např. nemáte dostatečně bezpečné heslo, vyplnil jste nevalidní email, formulář na to upozorní. Poté už se formulář chová jako v režimu controll a po opravení chyb ve formuláři zmizí varování upozorňující na chybně vyplněný formulář.

Z formuláře se ještě nevolá přímo na Supabase, formulář volá na hook, který následně volá na supabase. V případě registrace se po odeslání formuláře v hooku zavolá Supabase funkce signUp, které se předá email a heslo. Poté se naplní vytvořený záznam v tabulce users, do kterého vložíme first_name a last_name, které v tabulce od Supabase nemáme. Pokud všechno toto proběhne hook změní svůj stav na success a aplikace vás přesměruje na stránku s přihlášením.

2.4.2. Přihlášení

Pro přihlášení je opět napsaný formulář, který nekontroluje nic. Po odeslání se v hooku kontroluje, zda uživatel existuje a zda bylo zadáno správné heslo. Pokud tomu tak není vypíše formulář danou chybu. Pokud je přihlášení úspěšné do cookie se vloží JWT, který v sobě nese kromě jiného i id uživatele, který je přihlášený. Toto id se dá kdekoliv v aplikaci z cookie získat a několikrát je to použito. Poté vás aplikace přesměruje na úvodní stránku.

2.4.3. Protected routes

Protected routy se starají o to, aby uživatel, který není přihlášený nebo nemá oprávnění být na některých stránkách, se na ně nedostal. To, že se podle role a stavu přihlášení mění uživatelské rozhraní, ještě neznamená, že uživatel nemůže do url napsat „/admin“. Tomu, aby se na tuto stránku dostal musíme zabránit. Tento problém si můžeme rozdělit do 2 podkategorií.

2.4.3.1. Přihlášený / nepřihlášený uživatel

Když je uživatel nepřihlášený a snaží se dostat na některé stránky kam nemůže, tak v tomto případě uživatele můžeme přesměrovat ještě na straně serveru. V getServerSideProps si můžeme zkontrolovat, zda se v prohlížeči nachází potřebná cookie a pokud zde není klientovi rovnou pošleme přesměrování na přihlašovací stránku. Toto přesměrování se děje ještě před renderování uživatelského rozhraní.

2.4.3.2. Neoprávněný uživatel

Neoprávněný uživatel je někdo, kdo je již přihlášený, ale jeho role zakazuje přístup na některé stránky např.: uživatel nemůže přistupovat na admin stránku

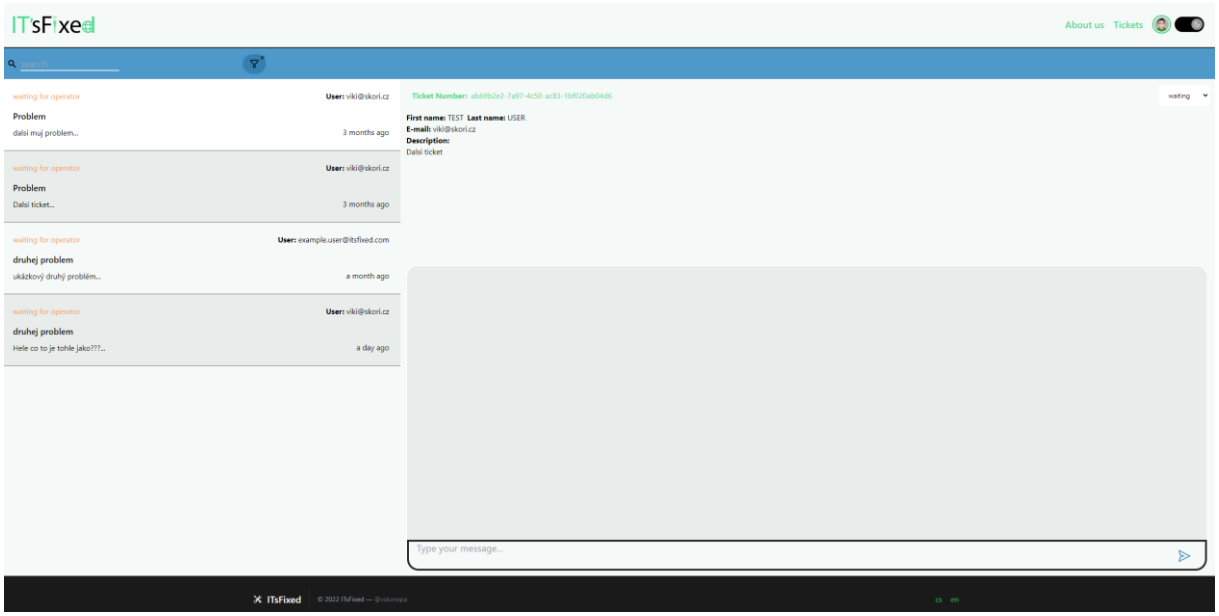
Na kontrolu tohoto problému můžeme využít user_id z cookie a zjistit pomocí dotazu na tabulku user_roles jeho roli. Bezpečnostní pravidla, ale v tomto případě povolují uživateli číst pouze záznam se svým user_id. Pokud bychom toto přesměrování chtěli opět vyřešit na serveru, museli bychom otevřít data v této tabulce pro všechny, což nechceme. Děje se to kvůli tomu, že se tento dotaz provádí na serveru, neposílá se s tokenem přihlášeného uživatele, ale pouze s veřejným Api klíčem. Musíme tedy přesměrování řešit na straně klienta. Zde už můžeme poslat dotaz na roli s tokenem přihlášeného uživatele a pokud jeho role neodpovídá podmínkám pro danou stránku, přesměruje ho. Jelikož dotaz opět probíhá v hooku, můžeme využít jeho stav loading k zobrazení pouze načítací stránky a až po jeho dokončení buď přesměrovat nebo stánku zobrazit.

2.4.4. Tickety

Tickety nesou hlavní funkci celé aplikace, skrze ně se problémy vytvářejí, řeší i ukončují. Na tickety se dá koukat ze dvou stran.

2.4.4.1. Tickety ze strany operátora

Obrázek 8 Ticket ze strany operátora



Zde je vytvořeno uživatelské rozhraní, viz obrázek č. 8, připravené na to, řešit jeden problém za druhým. Stránka je zde rozdělena na 2 části: ticket list a ticket detail. Obě části jsou na obrazovce zároveň, což umožňuje operátorovi po vyřešení ticketu ihned překliknout na další ticket a řešit.

2.4.4.1.1. Operátor ticket list

List jde filtrovat: podle typu problému a podle stavu ticketu. Ticket má 3 stavy: waiting, ongoing, done.

Operátor také může vyhledávat v popisu problému. Vyhledávání využívá funkci textsearch od supabase. Vyhledávání umí najít pouze jedno slovo.

List se nachází na levé straně obrazovky a je scrollovatelný pro jednoduché přepínání mezi tickety.

Data se zde stahují při prvním načtení stránky a je pro to opět vytvořen vlastní hook. Ten má 3 parametry, které se využívají k následnému filtrování. Při úvodním zavolání hooku parametry vkládáme jako prázdné s defaultními hodnotami. Pokud některou z těchto hodnot změníme, hook tuto změnu zachytí uloží stávající data do cache paměti pro případ, že by je někdy znovu potřeboval a provede refetch s novým parametrem.

Pro zajištění fungování v reálném čase je vytvořený subscription na tabulku tickets, aplikace si vytvoří spojení se supabase, která pokaždé, když se do tabulky přidá nový záznam, vyvolá callback funkci s novým záznamem. Jakmile tedy kdokoliv vytvoří nový ticket, operátor ho do několika sekund vidí a může řešit.

Zobrazované tickety se defaultně řadí od nejstarších k nejnovějším, aby se zabránilo hromadění nových ticketů nad staré, a tak by se některé tickety nemusely vyřešit.

2.4.4.1.2. Operátor ticket detail

Detail ticketu je pro operátora v pravé části obrazovky a mění se okamžitě po překliknutí na jiný ticket. V detailu jsou zobrazeny všechny informace o aktuálním ticketu a chat.

Po překliknutí na ticket se stáhnou všechny zprávy z daného ticketu a vloží se do pole. Toto pole je potom vypsáno do uživatelského rozhraní. Na jednu stranu se dávají zprávy s id, které se rovná vašemu a na druhou stranu zprávy, kde je id rozdílné tomu vašemu. Tím se zajistí, že vaše zprávy budou vždy vpravo. Pro chat se opět vytvoří subscription se Supabase. Díky tomu může chat fungovat v reálném čase a ani nevytěžuje síť, jelikož se vždy stáhne pouze nová zpráva a přidá se do pole všech ostatních zpráv.

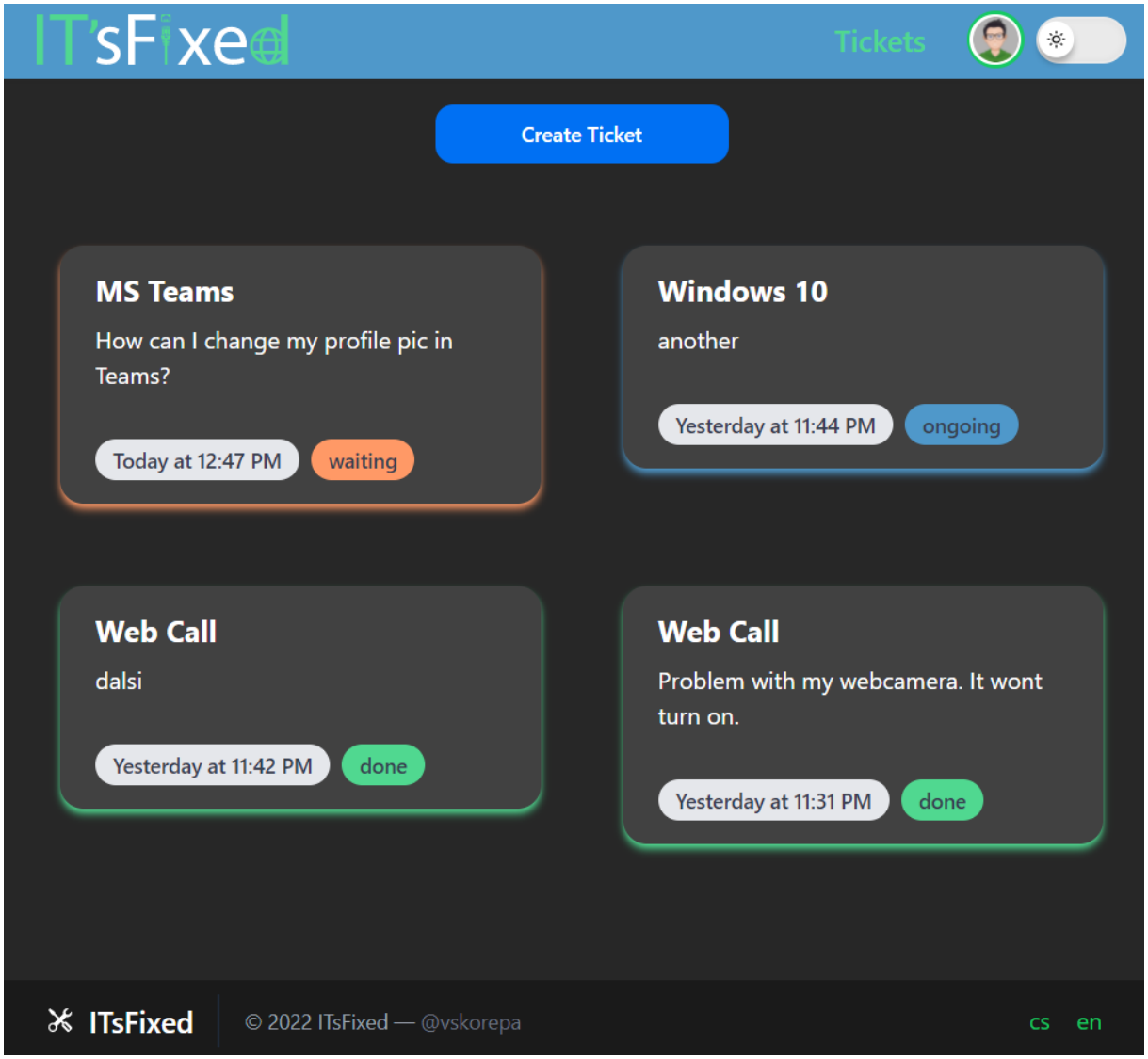
V detailu jde také změnit stav ticketu. Ten se mění pomocí výběru ze 3 možností vpravo nahoře. Při změně stavu aplikace provede mutaci na databázi opět pomocí vytvořeného hooku, která změní stav ticketu v databázi, a to vyvolá callback na subscription na tickety, tedy změní se stav ticketu a ticket zmizí z listu na levé straně.

Chat přebírá parametr disabled, který znefunkční odesílání zpráv v ticketu. Jakmile se Ticket přepne do stavu done, tento parametr se změní na true a do ticket již není možné zasílat zprávy.

2.4.4.2. Tickety ze strany uživatele

Uživatelské rozhraní pro uživatele bylo vytvořeno jinak než pro operátory. Hlavní rozdíl je, že uživatel má otevřený vždy jeden ticket a pro překliknutí na jiný se musí vrátit na seznam ticketů. Oproti operátorovi, který pro zrychlení jeho práce může překlikávat mezi tickety stále na jedné stránce. Viz obrázek č. 9.

Obrázek 9 ticket ze strany uživatele



2.4.5. Možnost povýšení na operátora

Pro uživatele byla naprogramována funkce, díky které mohou požádat o povýšení na operátora.

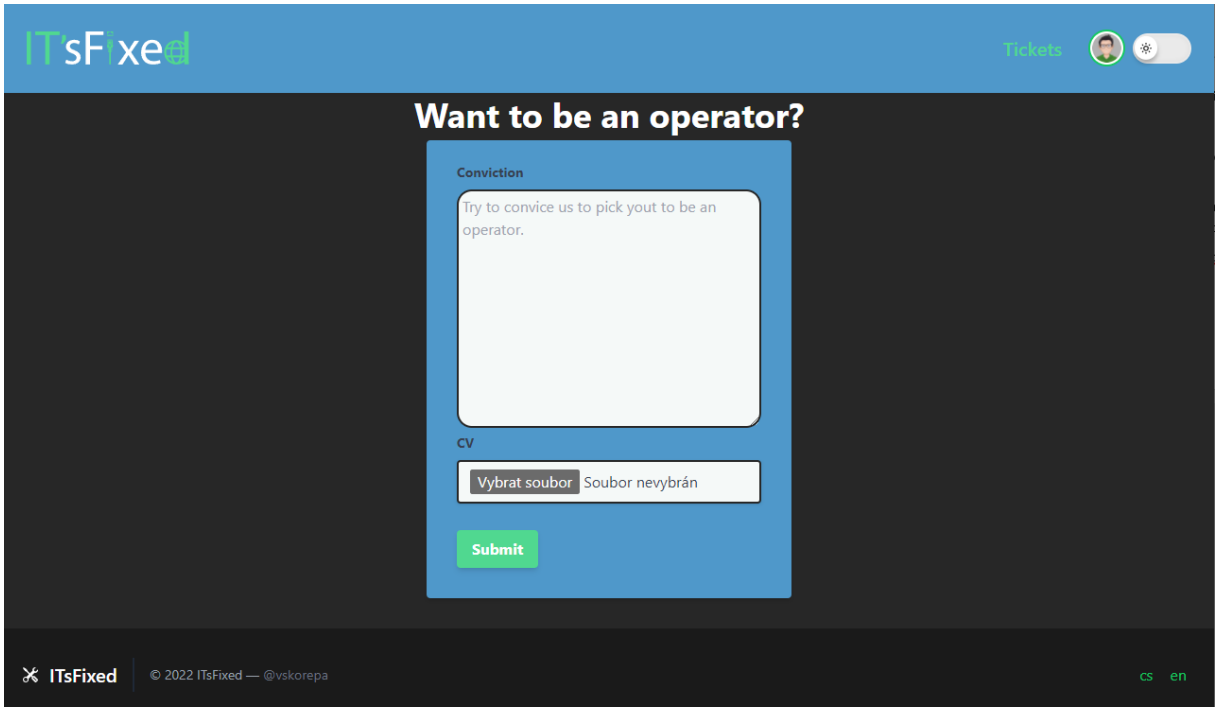
2.4.5.1. Formulář na odeslání žádosti o povýšení

Pokud chce být uživatel povýšen, musí vyplnit formulář na stránce /operator viz obrázek č. 10. Zde sepíše, proč je on kvalifikovaný k tomu být operátor a nahraje svůj životopis. U souboru je validováno, zda není větší než 20 MB a zda se jedná o PDF. Uživatel může odeslat formulář pouze jednou, a to díky kontrole unikátnosti sloupce user_id v tabulce.

PDF se uloží pod id uživatele do storage v supabase. Zde je pro to vytvořený tzv. bucket „cv-files“, na kterém je pravidlo, které dovoluje uživatelům vložit pouze soubor s názvem identickým jako jeho unikátní id.

Formulář po odeslání vytvoří záznam v tabulce operatorforms.

Obrázek 10 formulář na povýšení



2.4.5.2. Schválení / zamítnutí žádosti ze strany admina

Admin má na stránce /admin list všech žádostí a vidí u nich, kdy byly odeslány, text od uživatele a může si zobrazit životopis, který uživatel nahrál viz obrázek č. 11.

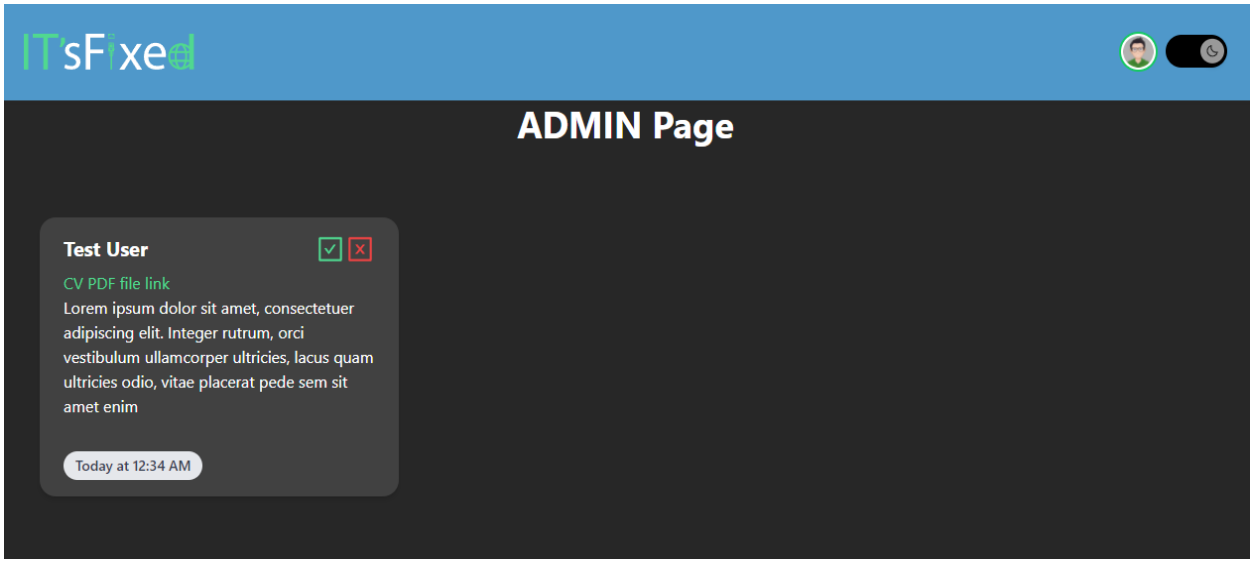
Na stránku se stáhnou všechny záznamy z tabulky operatorforms, které mají hodnotu true ve sloupci active. Tyto záznamy ještě nejsou nikým vyřízeny.

Jelikož se soubor v bucketu jmenuje stejně jako id uživatele, jednoduše se na něj odkazuje pomocí url adresy bucketu a id uživatele.

Poté, co admin vyhodnotí, jestli chce uživatele povýšit nebo nikoliv, může buď odkliknout ano jako povýšit nebo ne jako zamítnout.

Pokud se rozhodne že uživatele nechce povýšit, pouze se změní hodnota ve sloupci active na false a žádost už se nebude zobrazovat. Pokud chce uživatele povýšit, role uživatele, který žádost podal se změní z uživatele na operátora a poté se změní hodnota ve sloupci active na false.

Obrázek 11 list žádostí na povýšení



Závěr

V rámci projektu byla úspěšně naprogramovaná aplikace sloužící jako soběstačný helpdesk. Helpdesk s použitím moderních technologií jako je Next JS a Supabase, který může fungovat bez správce a bez údržby. Lidé se zde mohou přihlásit a vytvářet tickety. Lidé s praxí v IT si mohou zažádat o povýšení na operátora a pomáhat problémy řešit. I přes to, že projekt by mohl být vylepšován a rozšiřován o další užitečné funkce, jako by bylo např. využití mikrotransakcí nebo sdílení obrazovky tak je nyní otestován a ve funkční podobě na adrese: https://it-s-fixed.vercel.com.

Reference

[1] *Next.js by Vercel - The React Framework* [online]. [vid. 2022-03-14]. Dostupné z: https://nextjs.org

[2] *React – A JavaScript library for building user interfaces* [online]. [vid. 2022-03-14]. Dostupné z: https://reactjs.org/

[3] *JSX - What Is a JSX? & Introduction to Advanced* [online]. [vid. 2022-03-14]. Dostupné z: https://www.reactenlightenment.com/react-jsx/5.1.html

[4] The Open Source Firebase Alternative. *Supabase* [online]. [vid. 2022-03-14]. Dostupné z: https://supabase.com/

[5] *What is BaaS? | Backend-as-a-Service vs. serverless | Cloudflare* [online]. [vid. 2022-03-14]. Dostupné z: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/

[6] *PostgreSQL: About* [online]. [vid. 2022-03-21]. Dostupné z: https://www.postgresql.org/about/

[7] *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.* [online]. [vid. 2022-03-14]. Dostupné z: https://tailwindcss.com/

[8] supabase/examples/nextjs-slack-clone at master · supabase/supabase. *GitHub* [online]. [vid. 2022-03-14]. Dostupné z: https://github.com/supabase/supabase

Seznam obrázků