Delta – Střední škola informatiky a ekonomie s.r.o.
Ke Kamenci 151, Pardubice

# MATURITNÍ PROJEKT

## Systém pro výběr balíku

Leno, David
4.A
Informační technologie 18-20-M/01
2023/2024

# Zadání maturitního projektu z informatických předmětů

| | |
|---|---|
| Jméno a příjmení: | David Leno |
| Školní rok: | 2023/2024 |
| Třída: | 4.A |
| Téma práce: | Responzivní webová stránka pro nejlepší výběr balíku od dopravce podle parametrů zadaným uživatelem |

**Cíl a obsah práce:**

Cílem projektu je vytvořit responzivní webovou aplikaci, která pomůže vybrat nejvhodnější zásilkovou službu pro zasílání balíčků. Uživatel si navolí velikost balíčku, destinaci atp., a aplikace vyhodnotí vhodnou zásilkovou službu. Aplikace bude obsahovat i administrátorské rozhraní pro přidávání zásilkových služeb do systému. Admin bude moci přidávat i odebírat zásilkové služby a zadávat jejich parametry pro následný výpočet ceny pro balík.

Pro přihlášeného uživatele bude aplikaci nabízet i historii objednávek, možné předobjednáni, které se bude zobrazovat na stránce a možnost jeho zrušení nebo úplné objednání.

**Příklad použití aplikace:**

Uživatel si pro vybrání balíčku od dopravce zadá parametry jako např. váha, výška a šířka, odkud a kam je třeba balík doručit. Uživateli se zobrazí nejvhodnější dopravci pro balík, který nejlépe vyhovuje zadaným parametrům od uživatele.

**Základní technologie:**

- Firebase nebo jiná databáze
- React
- MUI – knihovna pro komponenty
- GraphQL

# Prohlášení

Prohlašují, že jsem práci vykonal samostatně, výhradně s použitím uvedené literatury.

V Pardubicích 21.03.2024

# Poděkování

Chtěl bych poděkovat Bc. Vlaďce Janů za odborné vedení projektu, za konzultace problémů aplikace při vývoji a za užitečné nápady, které mi pomohly projekt posunout vždy dále.

# Anotace

Cílem projektu je vytvořit uživatelsky příjemnou webovou aplikaci určenou pro vybrání zásilkové služby s vhodným balíkem dle parametrů zadaným uživatelem. Aplikace umožňuje správu nad databázi zásilkových služeb a balíků správcem webové aplikace. Uživateli umožňuje správu nad svým účtem (změna emailu, hesla) a vytvoření účtu, a jeho historií objednávek uložené uživatelem. Uživatel se bude moct přihlásit, registrovat a odhlásit.

# Klíčová slova

Zásilková služba, Balík, Webová aplikace, Firebase, Next.js, React

# Anotation

The aim of the project is to create a user-friendly web application designed to select a supplier with a suitable package according to the parameters specified by the user. The application allows management over the supplier and package database by the web application administrator. It will allow the user to manage over his account (change email, password, create account), and his order history stored by the user. The user will be able to log in, register and log out.

# Key words

Delivery service, Package, Web Application, Firebase, Next.js, React

# Obsah

Zadání maturitního projektu z informatických předmětů .....
Prohlášení.....
Poděkování.....
Anotace.....
Klíčová slova.....
Anotation .....
Key words.....
Obsah.....
Úvod.....8
Cílové skupiny.....8
1. Stránky pro uživatele.....9
1.2 Úvodní stránka .....9
2.2 Stránka pro admina .....10
3.2 Stránka uživatele .....12
4.2 Stránka pro registraci .....12
5.2 Stránka pro přihlášení.....13
6.2 Stránka pro vypočítání ceny.....13
2. Funkce .....15
2.1 Autentifikace .....15
2.1.1 Registrace.....15
2.1.2 Přihlášení .....17
2.2 Odhlášení.....17
2.3 Vhodná zásilková služba.....18
2.4 Vytvoření zásilkové služby, balíku a historie .....19
2.5 Odstranění zásilkové služby.....20
2.6 Úprava zásilkové služby.....20
2.7 Odstranění balíku.....21

2.8 Úprava balíku ..... 21
2.9 Mazání z historie ..... 22
2.10 Úprava historie ..... 22
2.11 Získání dat (zásilkové služby a jeho balík, historie) ..... 22
3 Použité technologie ..... 23
3.1 React ..... 23
3.2 Node.js ..... 23
3.3 TypeScript ..... 23
3.4 Graphql ..... 23
3.5 Firebase ..... 23
4. Architektura ..... 24
4.1 Databáze ..... 24
4.2 GraphQL ..... 25
4.3 Zabezpečení aplikace ..... 28
4.4 Use case diagram ..... 28
Závěr ..... 30
Reference ..... 31
Seznam obrázků ..... 32

# Úvod

Webová aplikace BingoBalik byla vytvořena pro lidi, kteří si chtějí zásilku poslat zásilkovou službou. Aplikace ušetří čas lidem, kteří by ji museli strávit návštěvou webů zásilkových služeb a vybrání vhodného podle toho, jakou zásilku chtějí poslat. Webová aplikace podle zadaných informací balíku uživatelem vyhodnotí vhodnou zásilkovou službu a nabídne uživateli, jestli chce objednat a přesměruje ho na oficiální web zásilkové služby nebo si jen objednávku uložit a pokračovat v objednávce až na stránce uživatele, kde má na výběr mezi správou uživatele nebo správou historie (mazání) nebo i objednání, kde ho přesměruje na stránku oficiální zásilkové služby, kde bude už přímo pokračovat v objednávce.
Samotná historie, co webová aplikace nabízí ,ušetří člověku spoustu času s hledáním té zásilky, kterou by na stránkách zásilkových služeb znovu objednával.

# Cílové skupiny

Cílovou skupinou jsou spíše mladí lidé, ale i lidé, kteří ocení funkce webové aplikace a místo nespočet návštěv webů zásilkových služeb pro vhodnou zásilku (balík), aplikace vypočítá a budete moct se rozhodnout podle celkové ceny, jestli je vyhovující dříve, než začnete proces objednávání a hledat cenu konkurenčních zásilkových služeb, jestli nejsou levnější.

# 1. Stránky pro uživatele

## 1.2 Úvodní stránka

Úvodní stránka obsahuje navigační panel, který vás přesměruje na jiné stránky a jeho položky (tlačítka) se mění dle toho, zda je uživatel přihlášený nebo není přihlášený a jestli je správce webové aplikace.
Panel nepřihlášeného uživatele (viz. Obrázek 1 Panel nepřihlášeného uživatele), přihlášeného uživatele (viz. Obrázek 2 Panel přihlášeného uživatele) a správce webu (viz. Obrázek 3 Panel správce webu).
Hlavní stránka obsahuje dvě tlačítka, které odkazují na stránku se správou účtu (viz. Obrázek 4 Přesměrování na stránku uživatele) a na stránku s vypočítáváním cen dle parametrů zadaných uživatelem (viz. Obrázek 5 Přesměrování na stránku pro vhodný balík).



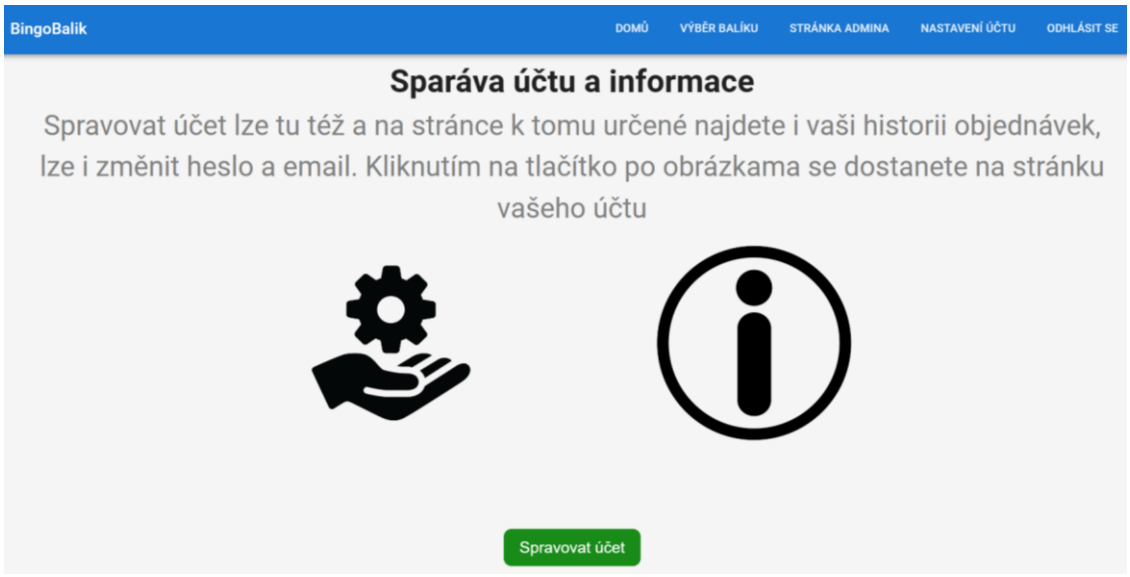

*Obrázek 1 Panel nepřihlášeného uživatele*



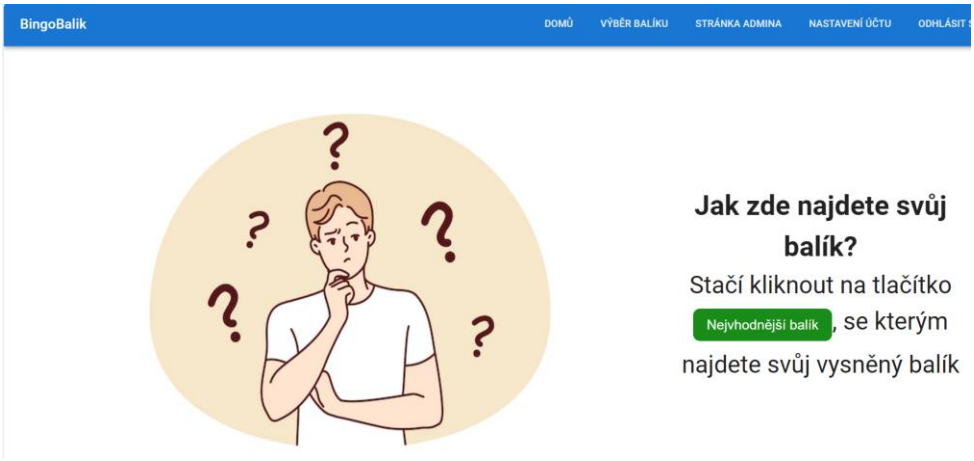

*Obrázek 2 Panel přihlášeného uživatele*




*Obrázek 3 Panel správce webu*



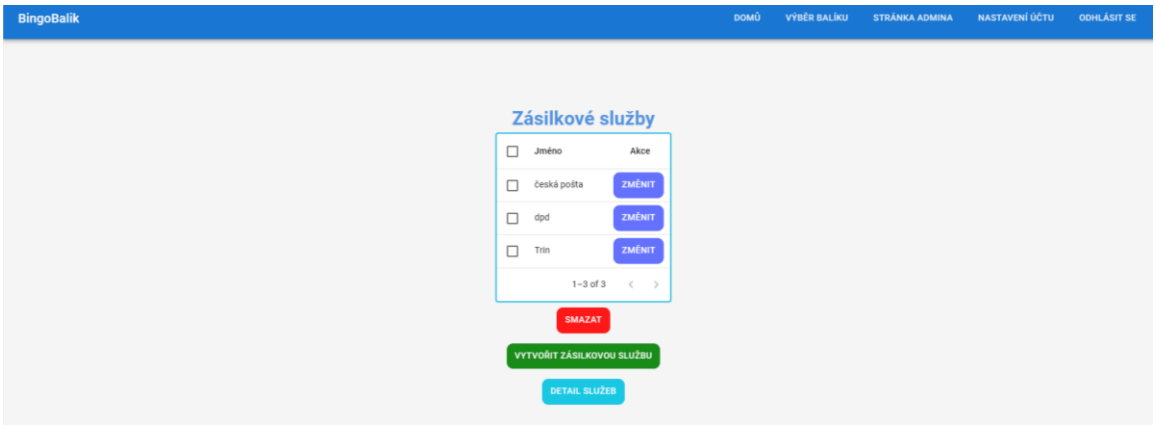

*Obrázek 4 Přesměrování na stránku uživatele*



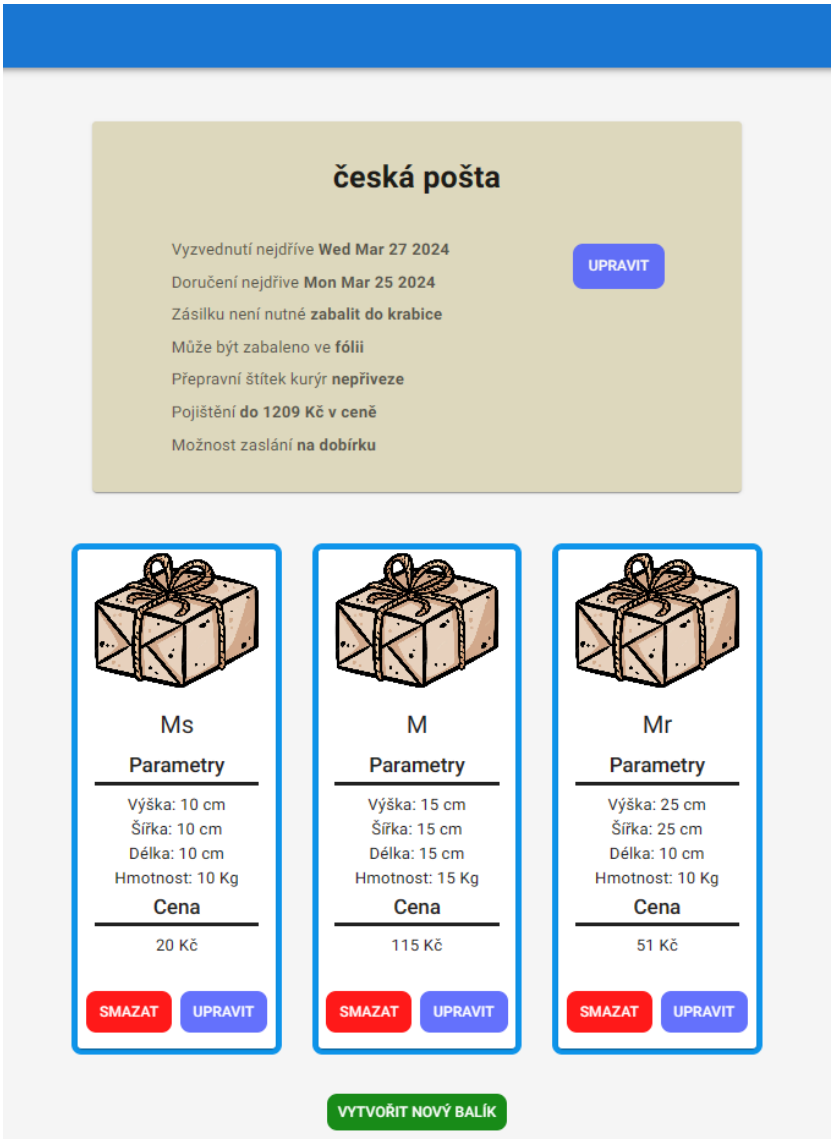

*Obrázek 5 Přesměrování na stránku pro vhodný balík*

## 2.2 Stránka pro admina

Stránka pro admina (viz. Obrázek 6 Stránka pro admina) umožňuje přidání zásilkové služby pomocí tlačítka „VYTVOŘIT ZÁSILKOVOU SLUŽBU“ nebo jí smazat pomocí tlačítka „SMAZAT“. Pro smazání musíte nejdříve kliknout na zaškrtávací políčko. Pro cílenější úpravu si musí administrátor vybrat zásilkovou službu kliknutím na zaškrtávací políčko, kde pak přímo může upravit, smazat jeho balíky a upravit samotnou vybranou zásilkovou službu (viz. Obrázek 7 Stránka admina pro specifickou úpravu).



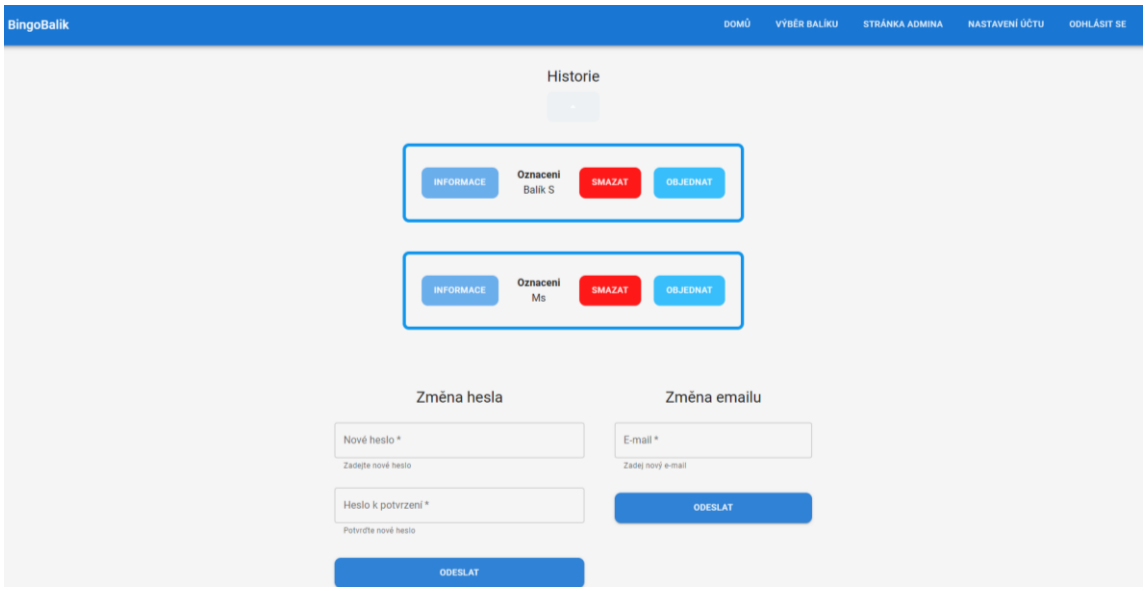

*Obrázek 6 Stránka pro admina*




*Obrázek 7 Stránka admina pro specifickou úpravu*

## 3.2 Stránka uživatele

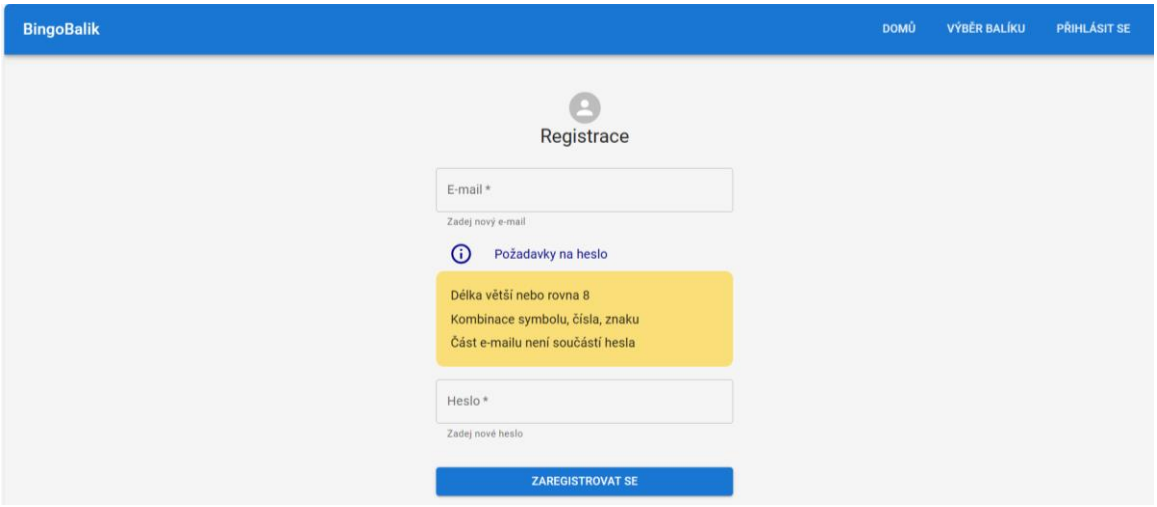
Stránka uživatele (viz. Obrázek 8 Stránka pro uživatele) umožňuje uživateli správu nad jeho účtem jako změnu emailu, hesla a manipulace s historií (mazání). Uživatel má možnost pokračovat v objednávce kliknutím na tlačítko „OBJEDNAT“ a zobrazit informace.




*Obrázek 8 Stránka pro uživatele*

## 4.2 Stránka pro registraci

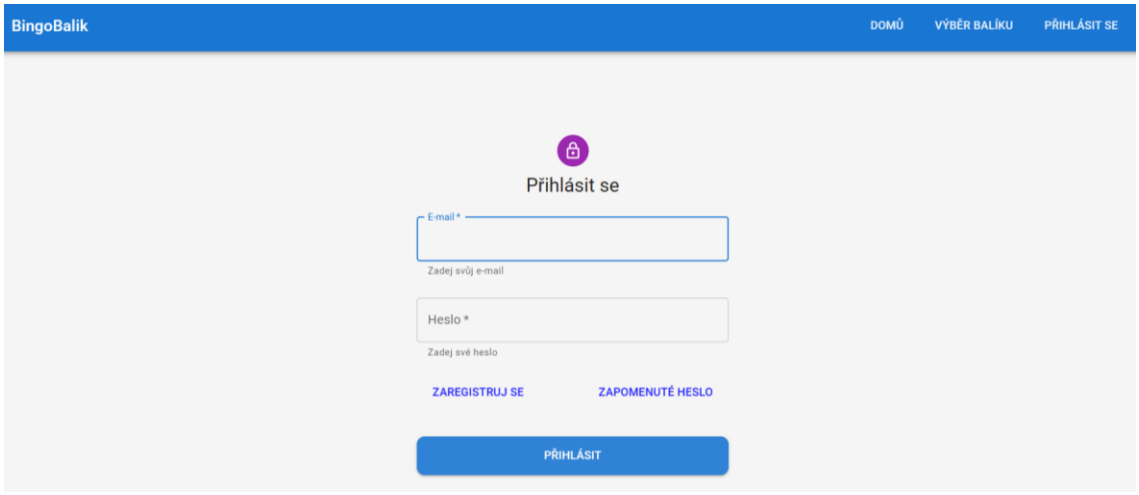
Stránka pro registraci (viz. Obrázek 9 Stránka pro registraci) obsahuje formulář, do kterého zadáte email a heslo. Heslo má požadavky, které musí splňovat. Na stránce je ikona „i“ pro zobrazení požadavků hesla, kde jsou napsány požadavky. Stránka též obsahuje navigační panel. Po úspěšné registraci vás přesměruje na hlavní stránku.




*Obrázek 9 Stránka pro registraci*

## 5.2 Stránka pro přihlášení

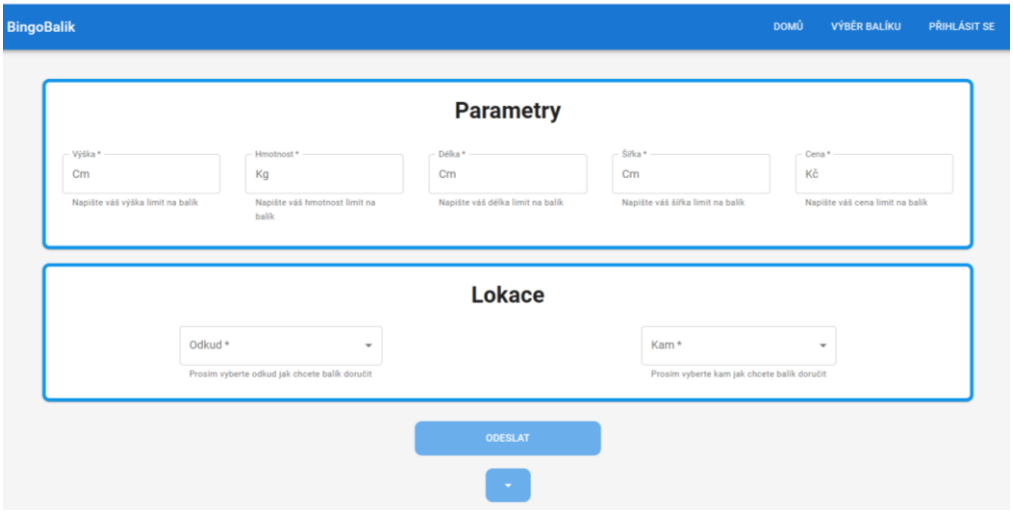
Stránka pro přihlášení (viz. Obrázek 10 Stránka pro přihlášení) obsahuje formulář pro přihlášení, kde zadáváte email a heslo. Po úspěšném přihlášení vás přesměruje na hlavní stránku. Stránka obsahuje navigační panel.



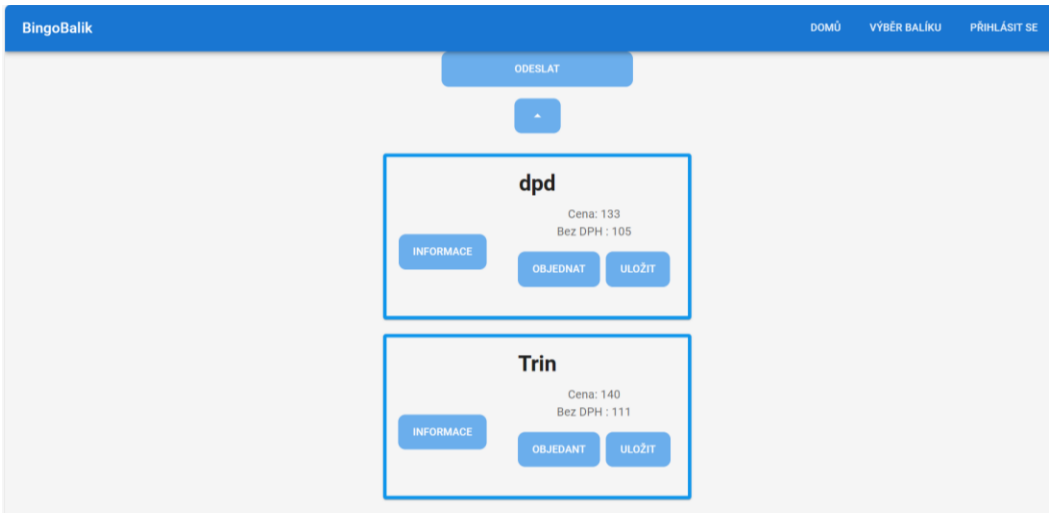

*Obrázek 10 Stránka pro přihlášení*

## 6.2 Stránka pro vypočítání ceny

Stránka obsahuje navigační panel a veliký formulář (viz. Obrázek 11 Formulář). Do velkého formuláře se zadají parametry balíku a možnosti vyzvednutí a dodání. Možnost vyzvednutí a dodání jsou na určené místo (osobně) nebo do pobočky (depo).
Po zadání informací uživatelem, je třeba kliknout pro zobrazení na tlačítko s šipkou a pomocí ní se karty i schovávají a zobrazí vhodné zásilkové služby s vhodným balíkem. Karty (viz. Obrázek 12 Karty) zobrazené po zadání umožňuji uživateli si zobrazit detail (tlačítko „INFORMACE“), objednat (tlačítko „OBJEDNAT“), které po kliknutí přesměruje uživatele na oficiální stránky zásilkové služby nebo si ho uložit do historie (tlačítko „ULOŽIT“).




*Obrázek 11 Formulář*




*Obrázek 12 Karty*

# 2. Funkce

## 2.1 Autentifikace

### 2.1.1 Registrace

Na obrázku (viz. Obrázek 13 Průběh registrace) je znázorněno, jak registrace funguje. Pošlou se data z formuláře, ale nejdříve, než se zadaná data pošlou do firebase authentication, tak se provede základní validace (viz. Obrázek 14 Funkce pro validaci). Při validaci se kontrolují zadaná data k novému účtu a to email a heslo, u kterého se kontroluje formát dle požadavků (viz. Obrázek 15 Požadavky na heslo).
Uživateli o chybách spojené s registrací se dává vědět ve dvou formách, a to pomocí alert komponenty (viz. Obrázek 16 Alert) nebo podle zabarvení pole pro zadávaní dat do červena s chybou po vyplnění formuláře (viz. Obrázek 17 Zabarvení pole s chybou).
Pokud bude vše validní, tak se pošlou data pro registraci do firebase authentication (viz. Obrázek 18 Poslání dat do firebase authentication) a zde když nastane chyba při validaci, tak uživateli zobrazí chybu už ve dvou zmíněných formách.
Po úspěšné validaci přesměruje uživatele na hlavní stránku s registrovaným účtem.




*Obrázek 13 Průběh registrace*

```
if (data.em.length === 0) {
  return { where: 'email', errMesage: 'E-mail nebyl poskytnut' };
}

if (data.pass.length === 0) {
  return { where: 'password', errMesage: 'Nebylo poskytnuto heslo
}

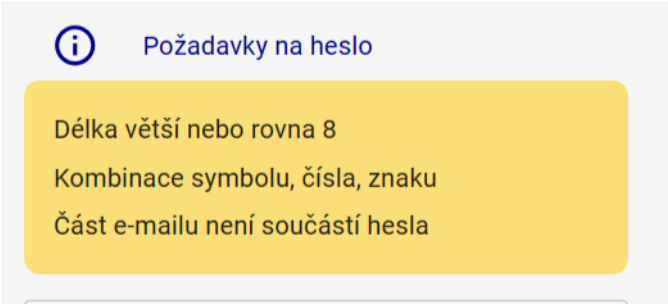
if (data.pass.length < 8) {
  return { where: 'password', errMesage: 'Délka hesla je malá' };
}

if (!hasSymbol || !hasNumber) {
  return {
    where: 'password',
    errMesage: 'Heslo není kombinací znaků, čísel a symbolů      ',
  };
}

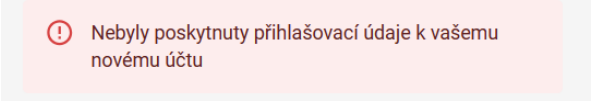
const getPartEmIndex = data.em.indexOf('@');

if (
  password.includes(data.em.slice(0, getPartEmIndex)) &&
  getPartEmIndex > 0
) {
  return {
    where: 'password',
    errMesage: 'Heslo obsahuje e-mailový účet      ',
  };
}
```

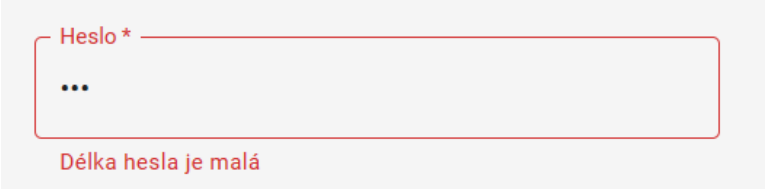
*Obrázek 14 Funkce pro validaci*




*Obrázek 15 Požadavky na heslo*




*Obrázek 16 Alert*




*Obrázek 17 Zabarvení pole s chybou*

```
try {
  await authUtils.register(data.email, data.password);
  await router.push('../../');
} catch (error) {
  const err = error as FirebaseError;
  switch (err.code) {
    case 'auth/email-already-in-use': {
      errSetter.errEmail.set('E-mail používá jiný uživatel');
      break;
    }
    case 'auth/invalid-email': {
      errSetter.errEmail.set('E-mail není valid');
      break;
    }
    default: {
      SetAlert(MyAlert('Registrace uživatelé nebyla úspěšná', 'error'));
    }
  }
}
```

*Obrázek 18 Poslání dat do firebase authentication*

### 2.1.2 Přihlášení

Uživatel vyplní formulář pro přihlášení a data z přihlašovacího formuláře se nejdříve zkontrolují, jestli informace byly zadány (viz. Obrázek 19 Průběh přihlášení), a pak se validuje email a heslo ze strany firebase authentication.
Ostatní validace se nechávají na firebase authentication a jenom se zachytávají chyby důležité k zobrazení uživateli a ostatní se zobrazí jenom jako přihlášení nebylo úspěšné (viz. Obrázek 19 Průběh přihlášení).
Po úspěšném přihlášení uživatele přesměruje na hlavní stránku.

```
if (data.email.length === 0) {
  errSetter.errEmail.set('E-mail nebylo zadáno');
  return;
}

if (data.password.length === 0) {
  errSetter.errPassword.set('Heslo nabylo zadáno');
  return;
}

try {
  await authUtils.login(data.email, data.password);
  await router.push('../../');
} catch (error) {
  const err = error as FirebaseError;
  handleErros(err.code, SetAlert, errSetter);
}
```

*Obrázek 19 Průběh přihlášení*

## 2.2 Odhlášení

Na odhlášení se používá metoda s firebase authentication (viz. Obrázek 20 logout metoda). Tato metoda uživatele odhlásí.

```
logout: async () => {
  await auth.signOut();
},
```

*Obrázek 20 logout metoda*

## 2.3 Vhodná zásilková služba

Získá všechny data z kolekce zásilkových služeb a spočítá případnou celkovou cenu. Následně vyfiltruje ty, které do představ ceny uživatelé nezapadají. Ty zbylé vyhovující filtruje na základě parametrů balíků. Z každé zásilkové služby porovnává jeho balíky mezi sebou a vybere jen jeden nejvhodnější a vrátí data se zásilkovou službou a nejvhodnějším balíkem a celkovou cenou (viz. Obrázek 21 Proces vybírání vhodných dat).




*Obrázek 21 Proces vybírání vhodných dat*

Spočítání ceny balíků (viz. Obrázek 22 Spočítání celkové ceny balíku).
Ceny balíků se zjistí, tak že získá všechny zásilkové služby z kolekce „Supplier“ a uloží si jeho balíky. Následně si z každé získané zásilkové služby uloží jeho id a celkovou cenu za způsob vyzvednutí a dodání.
Celková cena způsobu za vyzvednutí a dodání se dělí na 4 způsoby a každý má svoji cenu.

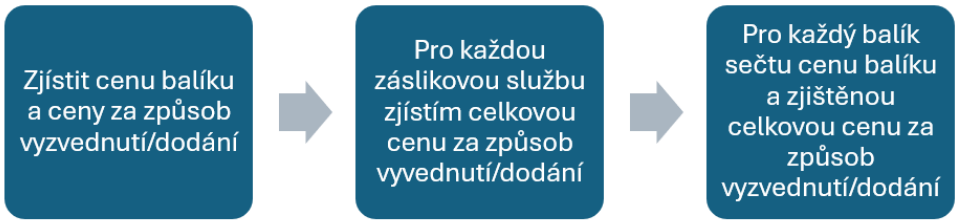
Osobně (na adresu) vyzvednutí-osobně (na adresu) dodání.

V depu (pobočka) vyzvednutí-do depa (pobočka) dodání.

Osobně (na adresu) vyzvednutí-do depa (pobočka) dodání.

V depu (pobočka) vyzvednutí-osobně (na adresu) dodání.

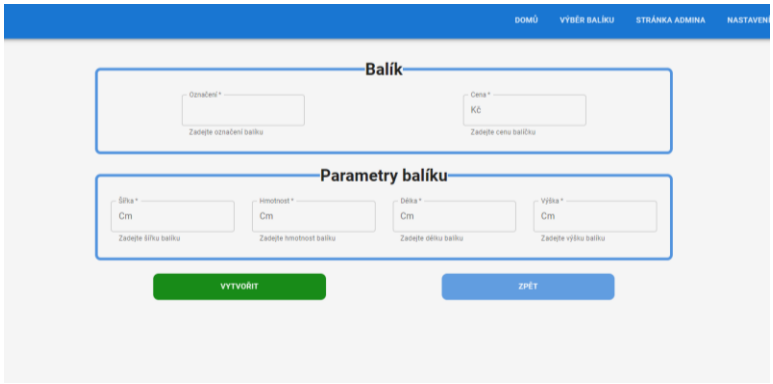
Po zjištění celkové ceny za dodání a vyzvednutí se ke každému balíku podle id zásilkové služby, který má uložený, sečte jeho cena s cenou za vyzvednutí a dodání podle zadaných dat od uživatele (uživatel si vybral například dodání osobně vyzvednutí osobně).
Následně vyfiltrujeme nevhodné balíky. Po filtraci se porovnávají balíky mezi sebou u každé zásilkové služby, a to podle parametrů zadané od uživatele a vrátí zásilkovou službu s vhodným balíkem, pokud bude vhodný.



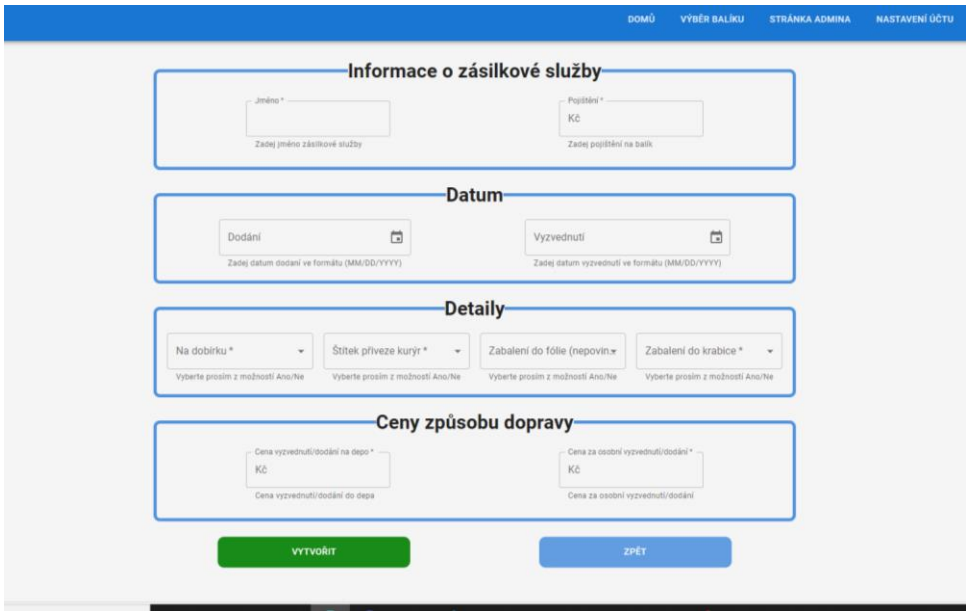

*Obrázek 22 Spočítání celkové ceny balíku*

## 2.4 Vytvoření zásilkové služby, balíku a historie

Z formuláře (viz. Obrázek 23 Formulář balíku, viz. Obrázek 24 Formulář zásilkové služby) získá data, ty předá mutaci pro vytvoření, která je nadefinovaná v objektu (resolvers), která příjme data z formuláře. Validace již částečně díky tvrdému typování kontroluje, jestli typ dat, co přišli jsou správného typu, dále pak se kontroluje jejich formát, který musí být splněn, jinak se vrátí jenom chyba.
Pokud je vše validní, tak se kontroluje už jen duplicita tím, že se požádá o dokument se zadaným jménem/označením a kontroluje se, jestli existuje. Pokud neexistuje, tak do kolekce se přidá dokument s daty z formuláře, jinak vrátí chybu.
K přidání nejvhodnějšího balíku do historie je třeba kontrolovat, zda už takový balík tento uživatel v historii nemá, jinak se přidá. K vytvoření získává podstatná data o samotné zásilkové službě a celkovou cenu i s označením balíku.




*Obrázek 23 Formulář balíku*



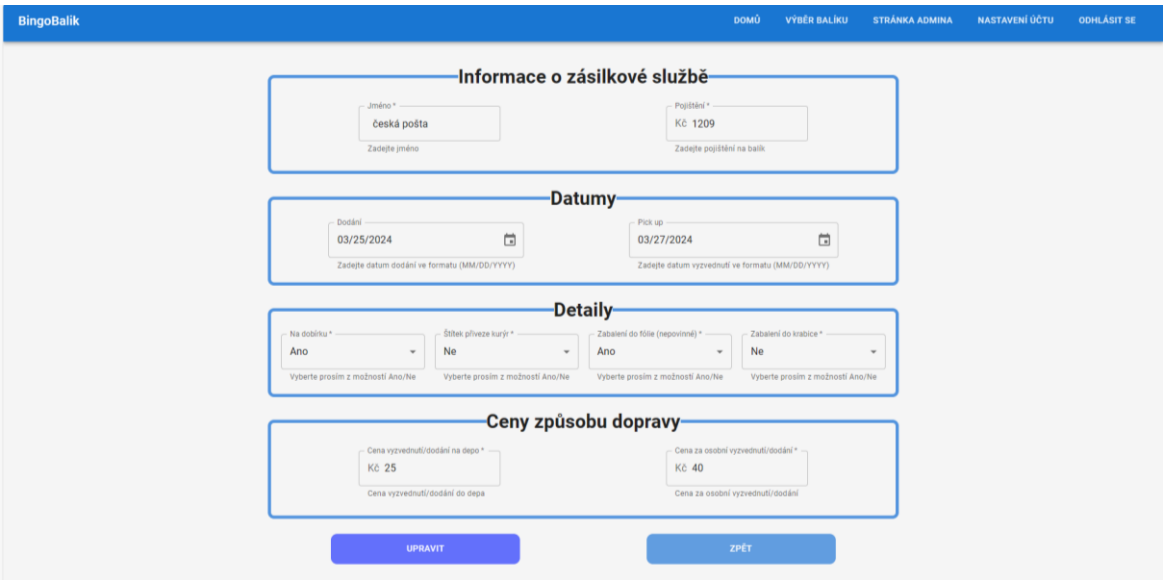

*Obrázek 24 Formulář zásilkové služby*

## 2.5 Odstranění zásilkové služby

Pro odstranění zásilkové služby je třeba pole s id a email aktuálního uživatele. Následně v asynchronní funkci dochází ke kontrole, zda je pole s id předáno. Kontroluje se oprávnění uživatele, pokud nemá uživatel oprávnění (není administrátor), mazání neproběhne a vrátí chybu, ale pokud je to v pořádku, tak při procházení pole s id si uloží dokument z firestore obsahující id a odstraní dokument z kolekce, a to pro každý id v poli. Též automaticky maže i zásilkovou službu z kolekce „Historie“.

## 2.6 Úprava zásilkové služby

Funkce přebírá id zásilkové služby a data z formuláře (viz. Obrázek 25 Formulář pro úpravu zásilkové služby). Kontroluje se, jestli jméno není duplicitní tím, že získá dokument zásilkové služby. Kontroluje typ, formát dat a oprávnění uživatele k úpravě. Pokud je vše v pořádku, tak podle předaného id najde dokument z kolekce, který toto id má a následně upraví dokument s daty z formuláře, jinak vrátí chybu.



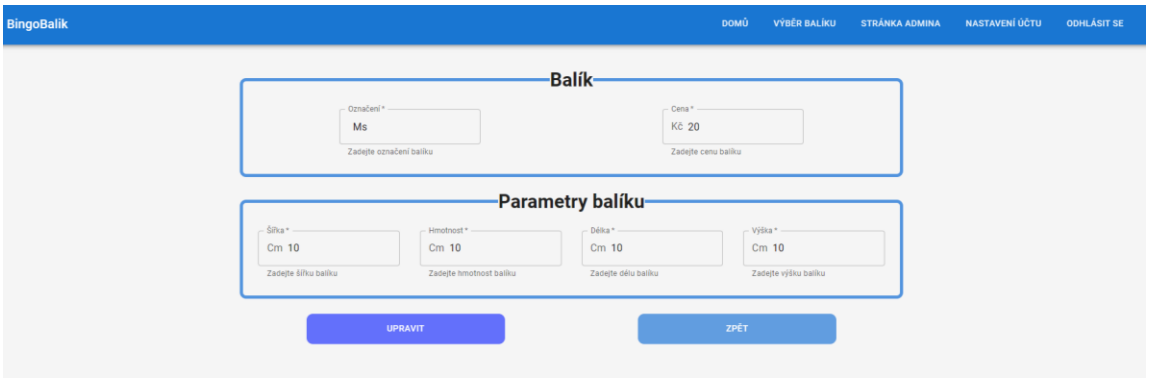

*Obrázek 25 Formulář pro úpravu zásilkové služby*

## 2.7 Odstranění balíku

Pro odstranění balíku je třeba id a email aktuálního uživatele. Následně v asynchronní funkci dochází ke kontrole, zda je id předáno. Kontroluje se oprávnění uživatele k odstranění, pokud nemá uživatel oprávnění (není administrátor), mazání neproběhne a vrátí chybu. Kontroluje se, zda balík s předaným id existuje, pokud balík s id neexistuje, mazání neproběhne a vrátí chybu. Pokud se předané id shoduje s id balíku z firestore, tak se vyfiltruje a upraví pole balíků v dokumentu.

## 2.8 Úprava balíku

Tato funkce potřebuje id balíku a data z formuláře (viz. Obrázek 26 Formulář pro úpravu balíku) od uživatele. Kontroluje se, zda data jsou ve správném formátu, id, duplicitní parametry, duplicitní označení a oprávnění uživatele k této činnosti. Pokud je vše v pořádku, tak se vezme balík s předaným id a přepíšou se jeho data na data z formuláře, jinak vrátí chybu.
Duplicitní název se kontroluje tím, že se získá dokument se zadaným názvem, ale ten aktuální nezahrnuje. U duplicitních parametrů se hledá balík, který má ty totéž u zásilkové služby.




*Obrázek 26 Formulář pro úpravu balíku*

## 2.9 Mazání z historie

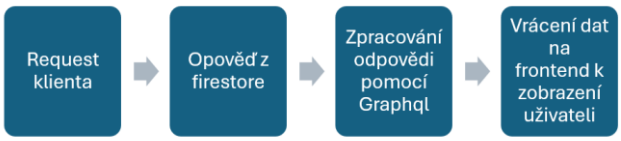
Metoda pro mazání z historie potřebuje id toho, co uživatel chce smazat. Kontroluje se, jestli je uživatel přihlášený a vybíráme dokument s id toho, co chce smazat z kolekce „History“ a zároveň dokument obsahující id přihlášeného uživatele. Pokud se takový najde smaže se.

## 2.10 Úprava historie

Volá se po každé úpravě balíku nebo zásilkové služby. Kontroluje se oprávnění uživatele. Přebírá nová data a kontroluje se, že opravdu data byly změněny, aby úprava dokumentu neprobíhala zbytečně. Když se data balíku, zásilkové služby nebo celková ceny za balík změnily, funkce aktualizuje data dokumentů v kolekci „Historie, aby historie uživatele byla konsistentní a zobrazovala aktuální data.

## 2.11 Získání dat (zásilkové služby a jeho balík, historie)

Na tuto funkci se použilo GraphQL. Tato funkce je definována v objektu (resolvers), kde v asynchronní metodě se dotazuje na všechny data z kolekce a uloží se všechna data z dokumentů, které kolekce má a metoda vrátí všechna data. K přístupu k datům a zobrazení je uživateli, slouží dotaz (query), která se skládá ze jména nadefinovaného dotazu a návratových hodnot. Průběh získávaní dat (viz. Obrázek 27 Získávání dat).




*Obrázek 27 Získávání dat*

# 3 Použité technologie

## 3.1 React

React je JavaScriptová knihovna pro vytváření uživatelských rozhraní, zejména pro webové aplikace. Byl vyvinut Facebookem a vydán jako open-source projekt. React umožňuje vývojářům vytvářet opakovaně použitelné komponenty uživatelského rozhraní a efektivně spravovat jejich stav.

## 3.2 Node.js

Node.js je softwarový systém navržený pro psaní vysoce škálovatelných internetových aplikací, především webových serverů. Díky spouštění kódu asynchronně se operace provádí zároveň.

## 3.3 TypeScript

TypeScript je open-source programovací jazyk, který je nadstavbou nad jazykem JavaScript a je vytvořený a spravovaný firmou Microsoft. TypeScript rozšiřuje JavaScript o statické typování a další atributy a je kompilován do jazyka JavaScript.

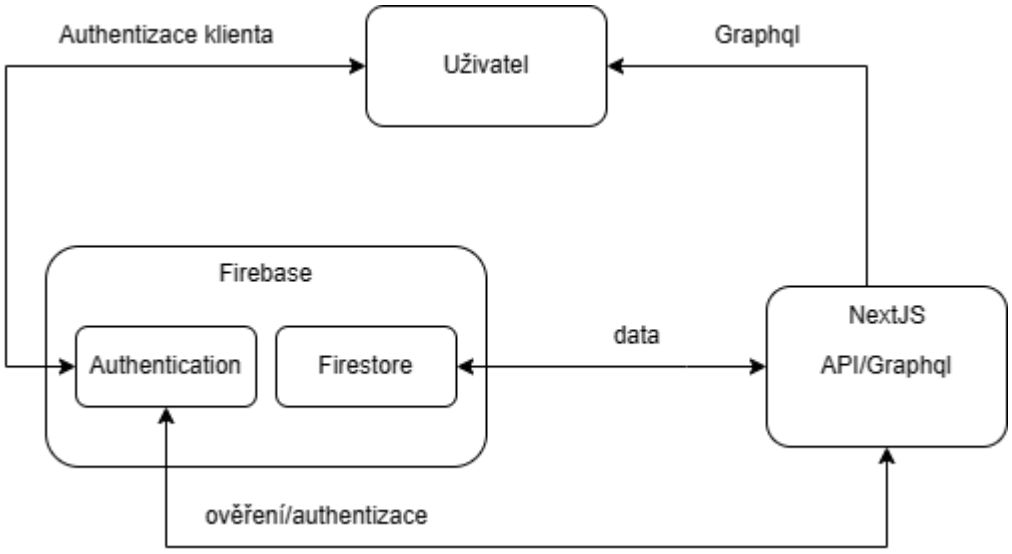
## 3.4 Graphql

Graphql je dotazovací jazyk, který je silně typovaný. Zde se musí specifikovat konkrétní data, která mají ze serverové částí přijít na klientskou.

## 3.5 Firebase

Cloudová společnost nabízející backend výpočetní služby, hostující databázi, správa uživatelských účtů. Je spravována společností Google.

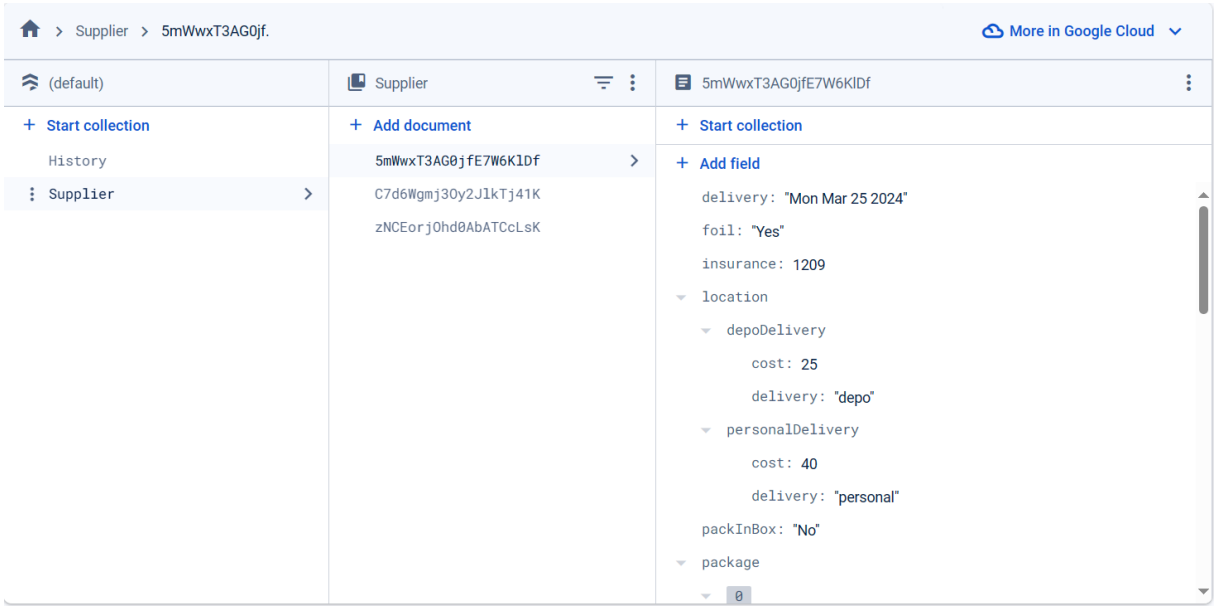
# 4. Architektura



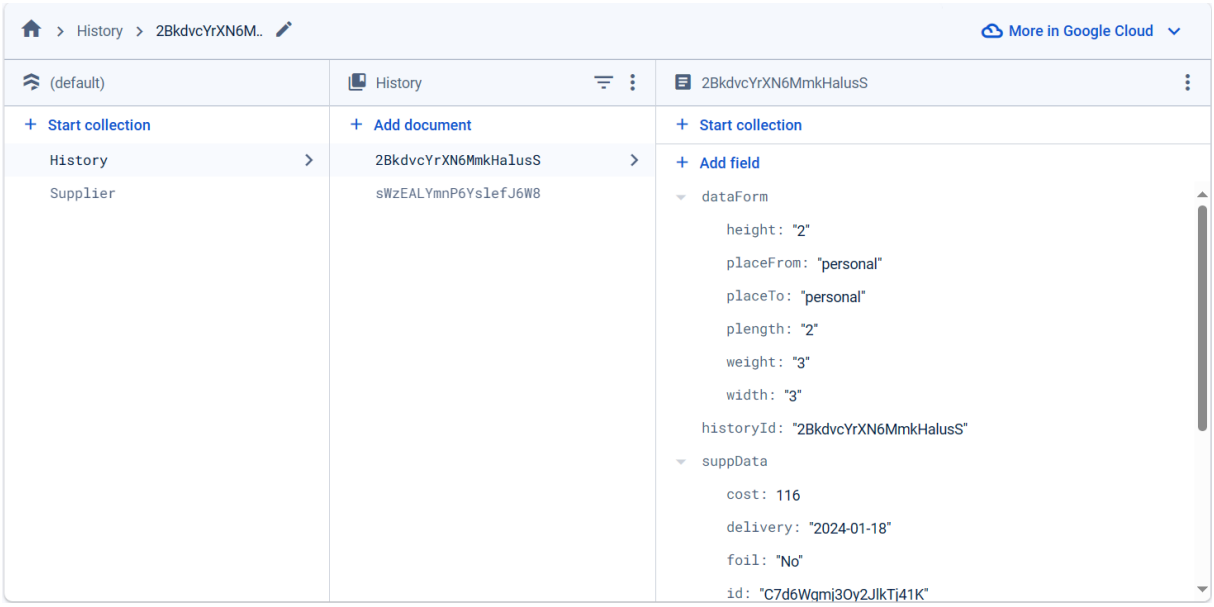

*Obrázek 28 Architektura*

## 4.1 Databáze

Jako databázi pro ukládání dat se používá firestore a to je NoSQL dokumentární databáze. Databáze, kterou aplikace používá obsahuje dvě kolekce, ve kterých jsou uložené dokumenty. V projektu se používá kolekce „History“ (viz. Obrázek 30 Kolekce History) a „Supplier“ (viz. Obrázek 29 Kolkece Supplier). Firestore umožňuje v jednotlivých dokumentech dělat další kolekce.




*Obrázek 29 Kolkece Supplier*




*Obrázek 30 Kolekce History*

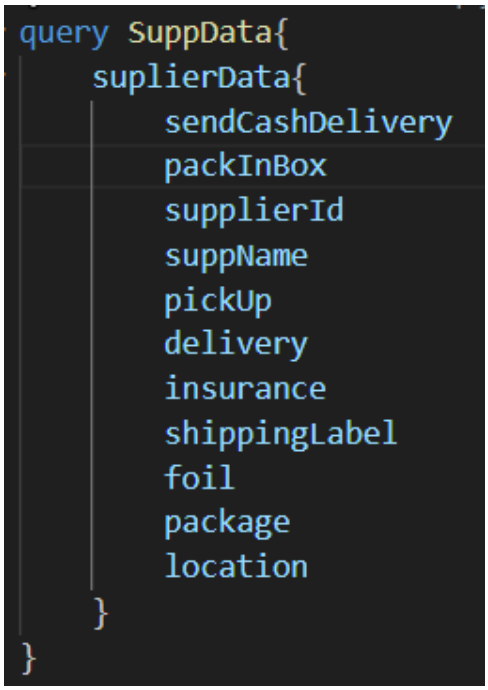
Jak je na obrázcích vidět každý dokument v kolekci má unikátní id a v dokumentech jsou velké možnosti, co se týče typů položek dokumentů (fields). V dokumentech používané pro ukládání a manipulaci dat jsou nejčastěji použity položky typu string, number, pole (array) a map (objekt). Právě pro balíky se použilo pole objektů v kolekci „Supplier“.

## 4.2 GraphQL

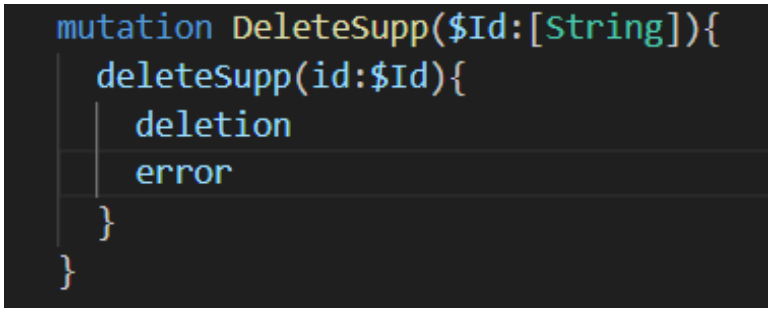
QraphQL je dobrý nástroj pro komunikaci mezi klientskou částí a částí serverovou. V projektu byla zvolena díky své jednoduchosti a také tvrdému typování, díky kterému právě kontrola na nevalidní data byla jednoduší.

V GraphQL se definují dotazy (viz. Obrázek 31 Query (dotaz) v GraphQL) a mutace (viz. Obrázek 32 Mutace v GraphQL), dotaz jenom data získává a mutace i manipuluje s daty. Například na funkci vyhodnocení nejlepší zásilkové služby s balíkem si potřebujeme nadefinovat mutaci pro použití na frontend a následně typy jak toho, co bude příjmat, tak i toho, co bude vracet a funkci, která provede vyhodnocení nejvhodnější zásilkové služby s balíkem.

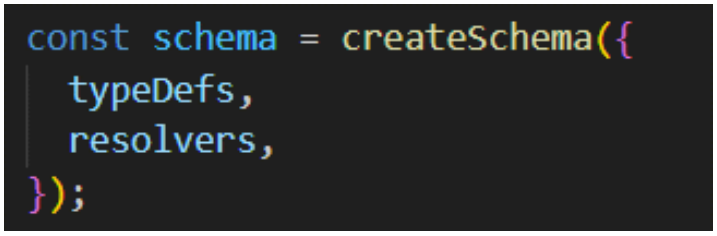
V projektu se používá na to funkce „createSchema (viz. Obrázek 33 Funkce pro definování typů a funkcí), která přebírá typy definující strukturu GraphQL API (typeDefs) a objekt (resolvers), které funkcionálně definují dotazy a mutace (viz. Obrázek 34 Nadefinované dotazy a mutace) definované v definicích typů (viz. Obrázek 35 Definice typů GraphQL).




*Obrázek 31 Query (dotaz) v GraphQL*




*Obrázek 32 Mutace v GrapQL*




*Obrázek 33 Funkce pro definování typů a funkcí*

```
const resolvers = {
  Query: {
    suplierData: async (…
    },
    historyUserData: async (…
    },
  },
  Mutation: {
    // vhodny balik resolver
    BingoSupPac: async (…
    },
    // web mutation…
    AddHistory: async (…
    },
    // eslint-disable-next-line complexity
    PackageToFirestore: async (…
    },
    SupplierToFirestore: async (…
    },
    // update
    // eslint-disable-next-line complexity
  },
};
```

*Obrázek 34 Nadefinované dotazy a mutace*

```
const typeDefs = gql`
  type Query {
    suplierData: [QuerySuppD!]!
    historyUserData: [QueryHistoryData!]!
  }

  type Mutation {
    BingoSupPac(
      width: Int!
      weight: Int!
      height: Int!
      Plength: Int!
      where: String!
      fromWhere: String!
      cost: Int!
    ): SuitValue
    AddHistory(uId: String!, data: String!): HistoryMessage
  }

  input DataUpdateSupp {
    sendCashDelivery: String
    packInBox: String
    suppName: String
    pickUp: String
    delivery: String
    insurance: Int
    shippingLabel: String
    foil: String
    supplierId: String
  }
```

*Obrázek 35 Definice typů GraphQL*

## 4.3 Zabezpečení aplikace

Se zabezpečením se právě pojí i použití GraphQL, která po každém požadavku na server extrahuje autorizační hlavičku z požadavku, ověří token uživatele a přidá ověřeného uživatele do kontextu, a to provádí funkce „context“ (viz. Obrázek 36 Funkce pro ověření uživatele („context“)). Po tomto ověření bude mít uživatel svůj bearer token (viz. Obrázek 37 Bearer token).
Bearer token se používá k ověřování a autorizaci a používá se ve webových aplikacích a rozhraních API k uchování uživatelských pověření a oprávnění pro požadavky a přístupu.

```
export default createYoga({
  schema,
  graphqlEndpoint: '/api/
  context: async (context
    const auth = context.request.headers.get('authorization');
    console.log(auth);
    return {
      user: auth ? await verifyToken(auth) : undefined,
    } as Context;
  },
});
```

(property) YogaInitialContext.request: Request
An object describing the HTTP request.

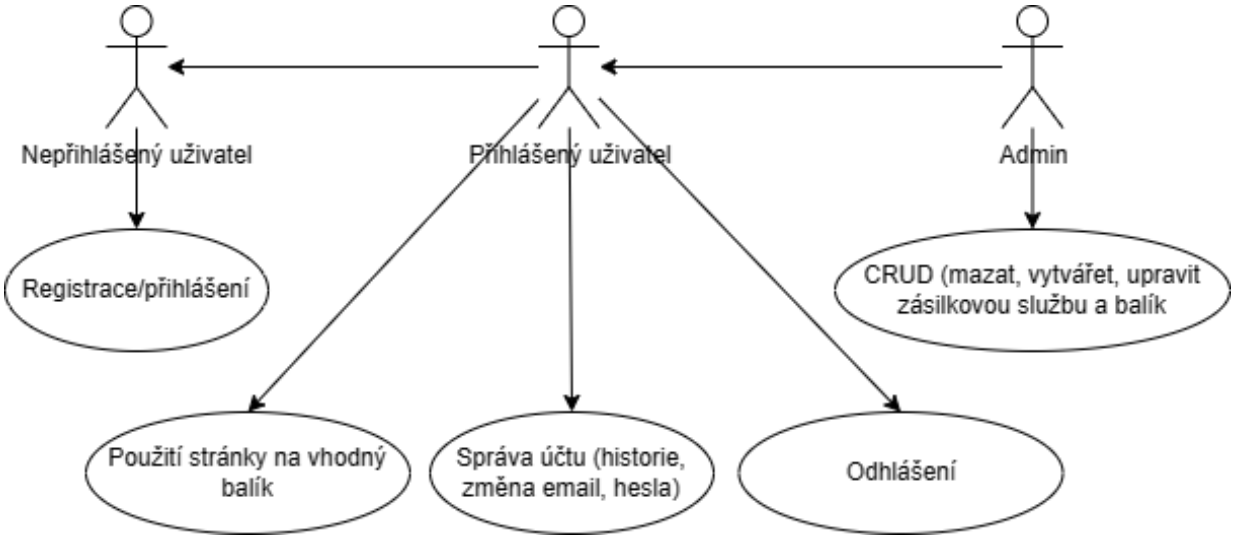
*Obrázek 36 Funkce pro ověření uživatele („context“)*

```
Bearer eyJhbGciOiJSUzI1NiIsImtpZCI6ImViY
```

*Obrázek 37 Bearer token*

## 4.4 Use case diagram

Use case diagram (viz. Obrázek 38 use case diagram) popisuje, jaké akce mohou jednotlivý uživatelé dělat. Nejvýše postavený je administrátor, který má možnost CRUD funkcí (vytvářet, číst, upravit, mazat), které manipulují s databází. Může spravovat svůj účet (změna emailu, hesla a historie) a použít stránku pro vhodný balík. Po administrátorovi následuje přihlášený uživatel, který ztrácí možnosti spojené s manipulací databáze a pak nepřihlášený uživatel, který má možnost přihlášení, registrace a použití stránky pro vhodný balík.




*Obrázek 38 use case diagram*

# Závěr

Cílem tohoto projektu bylo vytvořit webovou aplikaci pro vyhodnocení nejvhodnější zásilkové služby dle parametrů balíku zadané uživatelem. Tento cíl aplikace splňuje a všechny důležité funkce fungují.
Některé funkce, které jsem zamýšlel implementovány nejsou, ale ty důležité funkce aplikace má a fungují. Uživatelé mohou spravovat svůj účet a historii a odhlásit se, nepřihlášený se může registrovat a též může použít stránku s vyhodnocením nejvhodnější zásilkové služby, který má vhodný balík dle parametrů zadaným uživatelem, ale nemůže si zásilkovou službu s balíkem uložit do historie.
Administrátorské rozhraní pro správce je též funkční, mohou spravovat z webové aplikace databázi, která obsahuje zásilkové služby a jejich balíky.
Při vývoji jsem se mnohé naučil a programovat s vybranými technologiemi bylo příjemné.

# Reference


1. Node.js — Introduction to Node.js [online]. [cit. 2024-03-25]. Dostupné z: https://nodejs.org/en/learn/getting-started/introduction-to-nodejs
2. What is GraphQL? | Hygraph [online]. [cit. 2024-03-25]. Dostupné z: https://hygraph.com/learn/graphql
3. TypeScript: JavaScript With Syntax For Types. [online]. [cit. 2024-03-25]. Dostupné z: https://www.typescriptlang.oLekce 1 - Úvod do React [online]. [cit. 2024-03-25]. Dostupné z: https://www.itnetwork.cz/javascript/react/zaklady/uvod-do-react
4. Firebase Documentation [online]. [cit. 2024-03-25]. Dostupné z: https://firebase.google.com/docs
5. What is Bearer Token (An Ultimate Guide) [online]. [cit. 2024-03-25]. Dostupné z: https://apidog.com/articles/what-is-bearer-token/
6. Lekce 1 - Úvod do TypeScriptu [online]. [cit. 2024-03-25]. Dostupné z: https://www.itnetwork.cz/javascript/typescript/uvod-do-typescriptu
7. TypeScript Introduction [online]. [cit. 2024-03-25]. Dostupné z: https://www.w3schools.com/typescript/typescript_intro.php
8. Queries and Mutations [online]. [cit. 2024-03-25]. Dostupné z: https://graphql.org/learn/queries/
9. Next.js by Vercel – The React Framework [online]. [cit. 2024-03-25]. Dostupné z: https://nextjs.org/
10. Firebase Authentication [online]. [cit. 2024-03-25]. Dostupné z: https://firebase.google.com/docs/auth

# Seznam obrázků

Obrázek 1 Panel nepřihlášeného uživatele ..... 9
Obrázek 2 Panel přihlášeného uživatele ..... 9
Obrázek 3 Panel správce webu ..... 9
Obrázek 4 Přesměrování na stránku uživatele ..... 10
Obrázek 5 Přesměrování na stránku pro vhodný balík ..... 10
Obrázek 6 Stránka pro admina ..... 11
Obrázek 7 Stránka admina pro specifickou úpravu ..... 11
Obrázek 8 Stránka pro uživatele ..... 12
Obrázek 9 Stránka pro registraci ..... 12
Obrázek 10 Stránka pro přihlášení ..... 13
Obrázek 11 Formulář ..... 14
Obrázek 12 Karty ..... 14
Obrázek 13 Průběh registrace ..... 15
Obrázek 14 Funkce pro validaci ..... 15
Obrázek 15 Požadavky na heslo ..... 16
Obrázek 16 Alert ..... 16
Obrázek 17 Zabarvení pole s chybou ..... 16
Obrázek 18 Poslání dat do firebase authentication ..... 16
Obrázek 19 Průběh přihlášení ..... 17
Obrázek 20 logout metoda ..... 17
Obrázek 21 Proces vybírání vhodných dat ..... 18
Obrázek 22 Spočítání celkové ceny balíku ..... 19
Obrázek 23 Formulář balíku ..... 19
Obrázek 24 Formulář zásilkové služby ..... 20
Obrázek 25 Formulář pro úpravu zásilkové služby ..... 21
Obrázek 26 Formulář pro úpravu balíku ..... 22
Obrázek 27 Získávání dat ..... 22
Obrázek 28 Architektura ..... 24
Obrázek 29 Kolkece Supplier ..... 24
Obrázek 30 Kolekce History ..... 25
Obrázek 31 Query (dotaz) v GraphQL ..... 26
Obrázek 32 Mutace v GrapQL ..... 26
Obrázek 33 Funkce pro definování typů a funkcí ..... 26

Obrázek 34 Nadefinované dotazy a mutace ..... 27
Obrázek 35 Definice typů GraphQL ..... 27
Obrázek 36 Funkce pro ověření uživatele („context“) ..... 28
Obrázek 37 Bearer token ..... 28
Obrázek 38 use case diagram ..... 29