Úvod

Cílem práce je vytvoření webové aplikace pro dispečink taxislužby, která by měla zcela nahradit papírovou evidenci a tím výrazně zefektivnit získání přehledů objednávek, docházce řidičů, vozidlech a vazeb mezi nimi. Papírová evidence je časově náročná na vyhledávání a tvorbu přehledů objednávek pro řidiče za zadané období.

Zadávání veškerých dat obstarává dispečer, komunikující s řidiči vysílačkou a objednávky přijímá telefonicky v nepřetržitém směnném provozu.

Aplikace usnadňuje a zpřesňuje práci dispečera při velkém počtu řidičů k dispozici a vyřizování velkého množství příchozích objednávek v období poptávkové špičky, kdy dispečer je snadno zahltitelný a může dělat chyby.

Uživatelské rozhraní dispečerovi poskytuje přehled o stavu řidičů a stavu objednávek a řadí je do fronty k vyřízení. Výsledná aplikace by měla tuto agendu zcela nahradit.

Struktura práce začíná úvodem do řešené problematiky dispečinku. Následuje seznámení s použitými technologiemi a jejich výhodami. Dále pak vlastní řešení aplikace pro dané problémy. Poskytuji jednoduchý návod na použití pro uživatele. Závěrem bude shrnutí projektu a jeho vyhodnocení.

Motivace

Dispečerova práce v taxislužbě spočívá v komunikaci se zákazníky pomocí telefonu, či mobilní aplikace a s řidiči vozidel taxi pomocí vysílačky. Obvykle je přítomen dispečer v nepřetržitém provozu na směny.

Objednávky včasné i okamžité zapisuje do papírových sešitů. Po jejich vyřízení, tím je myšleno předání konkrétnímu řidiči k vyřízení zapisuje řidiče a vozidlo, kterým byla objednávka vyřízena.

Druhou agendou je docházka řidičů do práce. Vždy jsou k dispozici řidiči pracující na směny. Vedle nich jsou k dispozici řidiči, kteří nejsou na své směně, ale pouze čekají na zákazníky na stanovištích a nahlásili se dispečerovi jako záloha.

Řidiči v taxislužbě jsou koncesovaní živnostníci a od dispečinku si přidělené objednávky kupují. Proto je evidence všech pohybů nutná, aby mohl každý řidič jednou za měsíc dostat výpis jemu přidělených objednávek k vyrovnání. Tvorba přehledů a součtů je v papírové formě velmi zdlouhavá práce, kterou bude aplikace jednoduše automatizovat.

V poptávkové špičce dochází často k chybám na pravidlech přidělování objednávek, například může být čekající klient zapomenut, nebo řidič vyslán někam, kde už byla objednávka vyřízena.

Jako webová aplikace může být používána z libovolného místa a tím umožní vykonávat dispečerskou práci z domu za pomoci přenosné vysílačky, mobilního telefonu a zařízením s webovým prohlížečem, například tabletem. Tato skutečnost by měla vést ke zrušení pronajaté kanceláře a přechodu na práci z domu pro osoby se sníženou schopností pohybu a orientace.

Řešená problematika

Cílem práce je návrh a implementace webové aplikace pokrývající písemnou agendu dispečerské práce. Seznámení se a použití níže uvedených technologií pro tvorbu webové aplikace typu Single page aplikace. Implementovanou aplikaci otestovat na zkušební sadě dat v databázi.

Dispečink taxislužby je obsluhován v nepřetržitém směnném provozu a přijímá telefonické objednávky. Objednávky jsou řazeny do fronty podle jejich času k vyřízení. Ty okamžité jsou řazeny podle času jejich vzniku. Pokud se sejde v jeden čas okamžitá a včasná objednávka, pak včasná má vyšší prioritu. Uživatelské rozhraní by mělo tyto dvě fronty zobrazit a tím poskytnout dispečerovi možnost přidělovat objednávky bez chyb a spravedlivě podle výše zmíněných pravidel.

Řidiči jsou řazeni do fronty pro další přicházející objednávku, ve které mají přednost ti na své směně a pak ostatní k dispozici. Dále tvorba fronty musí zohledňovat stav řidiče, zda není mimo město, nemá pauzu, nebo je nepřítomen ve voze. Řidičům mimo směnu přiděluje dispečer objednávky pouze pokud ti ve směně jsou obsazeni a nemohou příchozí objednávku obsloužit.

Pro potřeby taxislužby je evidována i docházka dispečerů spolu s docházkou řidičů. Dále logování stavu řidičů během přítomnosti v práci pro přehled průběhu směny.

Použité technologie

PHP Framework Slim

Micro framework pro serverovou část aplikace, který je určen pro rychlou tvorbu malých webových aplikací, nebo API. Přijímá http dotazy, provede určité rutinní akce, obvykle práce s databází a vrátí http odpověď. Díky malému množství zdrojového kódu je v porovnání s klasickými PHP frameworky velmi odlehčený a rychlý. Poskytuje minimální sadu nástrojů pro serverovou část webové aplikace založené na klient-server architektuře. Slim jsem porovnal s jinými volně dostupnými a zjistil jsem, že pro mou aplikaci nenabízí význačně vhodnější možnosti. [1]

Slim Framework používá architekturu REST, tudíž podporuje všechny http metody (GET,POST,PUT,DELETE). Pomocí middleware vrstev lze aplikovat různě obecná pravidla a lze tak přesně upravit každou http odpověď.

Databáze MySQL

MySQL je jedna z nejpoužívanějších a nejoblíbenější open source relačních databází na světě. Databáze MySQL se díky osvědčené výkonnosti, spolehlivosti a snadnému používání stala volbou číslo jedna pro webové aplikace. [2]

Pro svou snadnou implementovatelnost (lze jej instalovat na GNU/Linux, MS Windows, ale i další operační systémy), výkon a především díky tomu, že se jedná o volně šiřitelný software, má vysoký podíl na v současné době používaných databázích.

React

Jedná se o javascriptovou knihovnu pro tvorbu interaktivních uživatelských rozhraní, převážně single page aplikací. Umožňuje navrhnout interaktivní uživatelské rozhraní a vykreslovat komponenty podle vnitřního stavu aplikace v reálném čase.

Každá komponenta musí implementovat metodu render(), která na základě vnitřního stavu komponenty vrací to, co se má zobrazit. Dále je důležitý objekt props, zkratka slova properties, který je povinným parametrem pro konstruktor každé komponenty.

Díky logice založené na jazyce JavaScript bývá obvykle datový model na straně klienta udržován v pomocných objektech, které jsou předávány do komponent jako parametry props. V těchto modelových objektech se volají procedury na načtených datech, či jejich získávání a posílání na serverové api.

Komponenty lze skládat libovolně, obvykle se každá komponenta skládá z menších a každá má odpovědnost za vykreslení konkrétní části DOM. Po vytvoření celé struktury DOM, se nadále aktualizují pouze ta místa, která se změnila.[3]

Na ukázce kódu v obrázku č. 1 vidíme jednoduchou komponentu HelloText, která vrátí nadpis s textem, který je předán pomocí props. Dále zastřešující komponentu App2, která jí používá a předává jí hodnotu „Dobrý den komponento“. Výsledek příkladu je nadpis s vloženým textem.

```tsx
export interface MyProps {
  helloText: string;
}

export class HelloText extends React.Component<MyProps>{
  render() {
    return (
      <h1>{this.props.helloText}</h1>
    );
  }
}

export class App2 extends React.Component<MyProps>{
  render() {
    return (
      <div className="mainContainer">
        <HelloText helloText={"Dobry den komponento"} />
      </div>
    );
  }
}

ReactDOM.render( <App />,document.getElementById('root') as HTMLElement);

```

MobX

gjgjkgjk

TypeScript

Bootstrap

Parcel

SPA

Klient server webová aplikace

Mysql, diagram, komentář diagramu a entit,

React

MobX

Serverová část

MVC, api

diagram

Klientská část

api call, react, mobx, asynchroní a synchronní části, dom,

gui ukázky

Závěr

- není přihlašování
- možno dodefinovat další možné přehledy
- rozšiřitelnost
- znovupoužitelnost i pro jiné dispečinky taxislužeb podobné velikosti

Zdroje

1. https://www.slimframework.com/docs/
2. Mysql web
3. https://reactjs.org/docs/rendering-elements.html
4.

http://jecas.cz/spa

http://frontendinsights.com/connect-mobx-react-router/

https://mobx.js.org/best/store.html

https://tylermcginnis.com/courses/react-router/

Pomocné knihovny

Grafické rozvržení obrazovky

2. Popis použitých metod (prostředků, obvodů, algoritmů, apod. - s odkazy na literaturu), je nepřípustné opisovat texty či kopírovat obrázky z literatury
3. Vlastní řešení konkrétního problému - popis vlastních prací, výsledků, apod.
4. Shrnutí výsledků projektu a závěr (naznačení dalšího možného pokračování)
5. Použitá literatura

Osnova

- Použité technologie
- Klient server webová aplikace
- Serverová část
    - Databáze, jednotlivé entity a jejich použití
    - Api
    - Dto
- Klientská část
    - Hlavní komponenty
    - Pomocné objekty
    - Mobx a React Router
- Návod k použití aplikace
    - Spuštění
    - Pravidla taxislužby a fronty
- Přehledy