Софийски университет „св. Климент Охридски”

**Факултет по математика и информатика**

**КУРСОВА РАБОТА**

**ПО**

**Разработка на клиент-сървър (fullstack) приложения с Node.js + Express.js + React.js**

**Тема:**

*Shopping assistant*

**Изготовил:**

**Калоян Богданов Гешев,**

**Фак. № 45221**

**Проверил:**

**/ Траян Илиев /**

# Съдържание

1. [Въведение, описание и идея на проекта](#_67m7ia3jla52)
2. [Основни технологии използвани за изготвяне на проекта](#_jma9jksmpb79)
3. [По-важни спецификации на проекта](#_ivr18q5ck2pq)
4. [Линк към сорс кода на приложението](#_cj0rj4igadrf)
5. [Някои материали използвани за изготвянето на проекта](#_dhr7wbuvcdtl)

# 1. Въведение, описание и идея на проекта

На всеки човек му се е случвало, при пазаруване в даден магазин да не може да открие даден продукт. За да може този продукт да бъде открит, даден човек трябва да се обърне към персонала на магазина и те да му дадат валидна информация за местоположението на продукта. Много често се случва персоналът също да не знае къде се намира даден продукт. Това създава предпоставка за непредвидено обикаляне на магазина и излишна загуба на време от клиента. За хора, които имат даден списък с покупки и искат максимално бързо да си напазаруват, това би било доста изнервящо и мнението им, за магазина и подредбата на продуктите в него, е възможно да спадне.

Идеята на този проект е да предостави на потребителите от този тип, средство, чрез което те ще могат максимално бързо да направят планираната си обиколка в магазина, без излишно губене на време, търсене на помощ от служители или ненужни обиколки.

Използвайки приложението, изготвено в тази курсова работа, потребител може да:

1. Разглежда информация за определени брандове магазини като - име на магазина, адрес на магазина, местоположение чрез google maps, списък с продукти, налични в специфичен магазин.
2. Разглежда всички въведени продукти в базите данни за всички магазини или отделни такива.
3. Сортира отделни продукти по бранд на магазин, цена, ново добавени и т.н.
4. Търси налични продукти в приложението.
5. За даден продукт потребител може да разгледа - името на продукта, кратко описание (не е задължително да има), цената на продукта, местоположението на продукта в магазина, в който се намира.
6. Местоположението на продукт се визуализира на карта генерирана от приложението.
7. Даден продукт може да бъде добавен в списъка за пазаруване.
8. Потребител може да си направи акаунт и така да отключи различни опции в приложението, според ролята на акаунта му.
9. Даден продукт може да бъде добавен в любими, при положение, че потребител е влязъл с акаунта си в приложението.
10. Потребител може да започне да пазарува чрез приложението **(Една от основните функционалности на приложението)**
    1. Пазаруването се осъществява, чрез изготвяне на карта на магазина и показване на локациите, на продуктите в количката, които потребителят е добавил.
    2. Показват се два списъка - един за продукти, които все още не са взети от потребител и продукти, които вече са взети от потребител, докато обикаля магазина.
    3. При натискане на продукт от даден списък той отива автоматично в другия - невзети -> взети и обратно
    4. При всяка интеракция със списъците се регенерира най-къс път в магазина, по който потребителят може да се ръководи и да приключи пазаруването бързо и доволно
    5. Пътят се изобразява на генерираната карта, като се слага и номерация на следващите продукти, които предстои да бъдат взети от потребителя в магазина.
11. Потребители с роля **Admin**, **Shop owner**, **Shop manager**, могат да менажират различни части от приложението
    1. Админ - Може да менажира всичко (потребители, магазини, продукти)
    2. Притежател на магазин - Може да менажира всички магазини, които са класифицирани с бранда притежаван от този потребител. Може да менажира всички мениджъри на тези магазини. Може да менажира информацията за магазините и продуктите под притежавания бранд.
    3. Мениджър на магазин - Може да менажира всички магазини, на които е мениджър - Променя информацията за магазина и продуктите в него. Може да бъде мениджър на магазини само от един бранд.
12. Едитор на карта - Потребителите с по-висок достъп в приложението може да променят картата на даден магазин, чрез едитора за карти добавен в приложението. Чрез този едитор могат да се чертаят възлите в магазина и да се съединят, така че да направят различни пътища. Всяка карта може да бъде експортната или може директно да се импортне карта от потребител в едитора, след което тя ще бъде визуализирана. Картите се съхраняват в JSON формат.

# 2. Основни технологии използвани за изготвяне на проекта

## React with typescript

Проектът се базира на съвременният фреймуърк - React. Също така за предотвратяване на грешки е добавен и typescript компилатор, който работи идеално в колаборация с фреймуърка. Бърз и лесен фреймуърк за изграждане на уеб базирани приложения.

## Material-UI

Готова система с визуални компоненти, които доста улесняват изграждането на дадено уеб приложение с реакт. Намалява времето за визуален дизайн на приложението.

## React router dom

Библиотека, която дава силата да се навигира между различните view-та (страници) на приложението.

## Redux

Редукс е доста популярна библиотека, която улеснява управлението на отделните състояния на дадено приложение. Тази библиотека работи прекрасно с React и решава доста проблеми, относно споделяне на състоянието на приложението в различни нива от дървото с компоненти.

Почти цялото състояние на приложението е изнесено в хранилището, създадено от библиотеката, което позволява доста гъвкав достъп до него.

## Express JS

За направа на сървърната/отдалечената част(REST API-то) на приложението е използван фреймуърка express. Чрез него доста лесно се дефинират моделите на пътищата, които ще прихващат различни заявки към сървъра от визуалната част на приложението.

## MongoDB

Всяко приложение, което работи с обработката на данни, които се съхраняват постоянно, трябва да има база с данни. В случая е използвана MongoDB, която база работи прекрасно в колаборация с Express JS.

# 3. По-важни спецификации на проекта

За визуалната част проектът е изграден на базата на SPA архитектура, която е осъществена чрез React, React router dom и redux.

За отдалечената част са засегнати основни точки от програмният интерфейс на отдалечената част, чрез използването на expressJS и mongodb:

### Потребители

Чрез заявки от имплементираните сървиси на визуалната част, могат да се осигури различна информация от отдалечената част на приложението. Основно могат да се вземат всички потребители(или информация само за един по идентификатор) наведнъж от базите данни чрез GET заявка към сървъра, което се случва единствено при стартиране на приложението и тази информация се записва временно в хранилището на redux, така че да не се правят излишни заявки към сървъра.  
Даден нов потребител може да бъде добавен(регистриран) в базите данни, чрез POST заявка към сървъра.

Даден потребител може да бъде премахнат от базите данни чрез DELETE заявка към сървъра.

Информацията за даден потребител може да бъде сменена (ъпдейтната) в базите данни, чрез PUT заявка към сървъра.

Относно потребителския програмен интерфейс за отделните заявки има прикачен мидълуер за валидация на някои основни характеристики като - дали даден потребител има права (дадена роля) за извършване на заявката и дали генерираният от сървъра токен, при влизане в системата, е валиден.

### Магазини

Чрез заявки от сървисите може да се осигури различна информация от базите данни за магазините от базите данни.

Могат да се вземат всички магазини наведнъж(или информация само за един чрез идентификатор) от базите данни, чрез GET заявка към сървъра.

Може да се добави нов магазин към базите данни, чрез POST заявка.

Може да се изтрие даден магазин от базите данни, чрез DELETE заявка.

Може да се промени/ъпдейтне информация за даден магазин, чрез PUT заявка.  
За някои заявки при магазините отново има валидация, чрез мидълуер за правата на потребителят, т.к не всички заявки могат да се извършват от анонимни потребители.

### Брандове на магазини

Същото, както при магазините от предишната точка, но за отделни брандове магазини

### Продукти

Същото, както при магазините от предишната точка, но за продукти в магазините.

### 

### Как е организирана информацията (релации м/у отделните домейни)

Бранд магазин е в релация с отделни магазини от този бранд - Всеки бранд пази масив от идентификатори към магазините с този бранд.

Бранд магазин е в релация с отделни продукти - Всеки бранд пази масив от идентификатори към продуктите от този бранд.

Магазин е в релация само с един бранд магазин - Един магазин може да има само един единствен бранд (но всеки магазин от даден бранд е различен, защото например може да има различно местоположение).

Продукт е в релация с бранд магазин - Един продукт присъства само в един даден бранд магазин.

Потребител е в релация с продукт като любим/ продукт в кошница- Даден продукт може да присъства в масив от любими продукти на потребителя, със съответния си идентификатор. Продукт може да присъства в кошницата на потребителя, абстрактно изложена като масив, чрез идентификатора си в този масив.

### Карта и генериране на път в карта

Най-важната отличителна черта на проекта е картата, върху която се изобразява най-късият път в магазина с добавените в кошницата продукти от потребителя.

Самата карта е направена, чрез SVG(scalable vector graphics) поради лесното и интуитивно чертане на различни векторни елементи, чрез зададен подобен на XML/HTML синтаксис.

Относно начертаването на път в тази карта се ползва алгоритъм за намиране на най-къс път в граф - Дейкстра. Чрез алгоритъма се генерират отделните възли на магазина, които трябва да бъдат посетени за да се стигне до продуктите. Поради факта, че SVG е представено в двуизмерно пространство са използвани x, y координати за отделните възли на картата. Понеже знаем, че Дейкстра изисква и тегла на ребрата м/у възлите, те се пресмятат за всяка двойка възли и се запазват в техните данни.

### Едитор на картата

Едиторът на картата също е изграден чрез SVG, защото то позволява да се слуша за различни евенти от мишката - click, mousedown, mouseup и т.н. Точно поради това, е доста лесно да се изгради логика, която позволява добавяне, премахване и промяна на елементите от дадено SVG. Едиторът поддържа следните функции:

* Добавя възли в магазина, чрез добавяне на точка в картата
* Свързване на възли в картата, което чертае линия м/у две точки и съставя пътят, по който потребител може да се движи
* Изтриване на възли от картата - премахва дадена точка
* Местене на възли по картата - дава възможност за промяна на положението да дадена точка от картата
* Задаване на начална точка на картата - Ползва се за задаване на входа на магазина, от където генераторът на път да започне своят алгоритъм.

# 4. Линк към сорс кода на приложението

Линк към сорса - <https://github.com/alien1976/shopping-assistant-react>

# 5. Някои материали използвани за изготвянето на проекта

* React - <https://reactjs.org/>
* Redux toolkit - <https://redux-toolkit.js.org/>
* ExpressJS - <https://expressjs.com/>
* Mongodb - <https://www.mongodb.com/>
* MaterialUI - <https://material-ui.com/>