**Софийски университет „Св.Климент Охридски“**

**Факултет по математика и информатика**

**Документация**

Към проект „Full stack project“ на тема:

***„iStore“***

**Изготвили:**

**Георги Димов, ФН:71572**

**Валентин Георгиев, ФН 71563**

**Ивелин Тодоров, ФН 71547**

**Съдържание:**

# Кратко описание на проекта

# Функционални изисквания

# Нефункционални изисквания

# Използвани модули (референции )

Не-тривиалните аспекти на архитектурата и реализацията на системата

Инсталиране и конфигуриране на разработената система

Потребителска документация

Заключения

# Кратко описание на проекта

Проекта който пожелахме да разработим и представим е онлайн магазин пазаруване. Имплементиран е посредством REST модел-а. Както във всеки магазин ние имаме потребители , администратор ( управител на магазина) , продукти и кошница, в която да сторираме избраните от нас продукти. Имплементирали сме познатите функционалности, необходими за пълноценно и удобно онлайн пазаруване. Всеки от нас имплементира по 2 модела, както и по 2 или повече компонента.

# 

# Функционални изисквания

Имплементирали сме всички необходими функционалности, за да може един потребител да изпита удоволствие от пазаруването. След поредица от екипни сбирки, решихме да добавим следните фунционалности:

* **login**
* **logout**
* **register**
* **authentication**
* **admin can add products**
* **add product to the basket**
* **remove product from the basket**
* **add comments**
* **remove comments**
* **select between different addresses when making an order**
* **view products by category**
* **view basket**
* **view profile information**
* **add multiplе addresses**
* **view addresses**
* **checkout process**
* **order history (pending)**

# Нефункционални изисквания

Като цяло нашите нефункционални изисквания са насочени към сървъра , на който ще се стартира нашият сайт. Трябва да разчитаме на издръжливост, стабилност и надежност.

* **Load time to be lower than <0.5 sec**
* **Database to be bigger than 500 mb**
* **Big server disk space ( needed for more item pictures)**
* **Support big traffic**
* **Support all browsers**

# Използвани модули (референции )

* [react](https://www.npmjs.com/package/express) - основната реакт библиотека
* [react-dom](https://www.npmjs.com/package/react-dom) -
* [react-router](https://www.npmjs.com/package/react-router) - реакт модул за роутинга
* [super-agent](https://www.npmjs.com/package/react-scripts) - **client-side** HTTP request library
* [jsonwebtoken](https://www.npmjs.com/package/jsonwebtoken) - модул за аутентикация
* [mongoose](https://www.npmjs.com/package/react-scripts) - модул за млаб модели и заявки
* [body-parser](https://www.npmjs.com/package/react-scripts) - модул за получаване и изпращане на json заявки
* [cors](https://www.npmjs.com/package/react-scripts) - модул, позволява комуникация на back-end & front-end server-a
* [express](https://www.npmjs.com/package/react-scripts) - web framework for node js
* [immutability-helper](https://www.npmjs.com/package/react-scripts) - за сетване на state на обект
* [bcrypt-nodejs](https://www.npmjs.com/package/react-scripts) - енкриптиране на потребителските пароли
* [react-scripts](https://www.npmjs.com/package/react-scripts)
* [nodemon](https://www.npmjs.com/package/nodemon) - live server, който рестартите
* [morgan](https://www.npmjs.com/package/morgan) - logging requrest details

# Не-тривиалните аспекти на архитектурата и реализацията на системата

В общи линии сме се придържали към по тривиална концепция. Като по не тривиално можем да посочим Auth.js файла, който предоставя функционалност за аутентификация и предоставя данните на влезналия в системата потребител. Тази функционалност е изразена като един обикновен javascript class.

# Инсталиране и конфигуриране на системата

Инсталацията е доста лесна, семпла и бърза.Нека първо инсталираме зависимостите:

* **npm install**

След което можем да стартираме нашето приложение посредством следната команда:

* **npm start**

По подразбиране порт-а на front-end сървър-а е 3001, но при нужда той винаги може да бъде заменен с друг. Просто отидете в **scr** директирията и редактирайте **config.js** файла.

# Потребителска документация

Нека разгледаме нашата система от потребителска гледна точка. След като потребителя е влезнал в нашата система, той ще може да се докосне до едно страхотно изживяване, а именно онлайн пазаруването. След като е влезнал в системата потребителя може да разгледа и избере някой от наличните артикули. За да извърши поръчката си , потребителя трябва да мине през следните етапи:

* регистрация
* влизане в системата
* добавяне на актуален адрес
* селектиране на продукт
* добавяне на продукт в кошницата
* checkout
* селектиране на актуален адрес

# Заключение

При разработката на системата винаги гледахме да се събираме, докато програмираме. При целия процес не срещнахме големи трудности. Всеки един срещнат проблем сме го обсъждали и решавали с общи усилия. Едно от малкото неща които ни затрудни беше следния въпрос:

Как да извикаме дадена заявка в един компонент веднъж и нейния отговор да бъде предаден на всички деца ?

След кратко проучване в интернет стигнехме до следното решение - да предадем отговора на заявката екстендвайки this.props .