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:**

*Event Management System*

**Изготовили:**

**Ивайло Илиев, фн: 81402**

**Проверил:**

**/ Траян Илиев /**

**Николай Найденов, фн: 81565**

# Съдържание

1. **Въведение, описание и цел на приложението**
2. **Основни технологии, използвани за изготвяне на приложението**
3. **По-важни спецификации на приложението**
4. **Линк към source-кода на приложението**
5. **Използвани материали за изготване на приложението**

# 1. Въведение, описание и цел на проекта

Идеята на този проект е да предостави на крайните потребители възможност за менижиране на събития от всякакъв тип, както и да се създава статистика, в зависимост от предпочитанията на потрепителите към дадено събитие.

Системата предоставя функции, с които се улеснява значително разпространяването на дадено събитие, събирането на информация за него, както и средства за комуникация между потребителите, заявили участие в него.

Създаването на различни организации, към които могат да принадлежат много събития, е също значителна част от функционалността на приложението.

Функционалностите, предоставени от приложението са:

1. Вход/Регистрация на потребител
2. Създаване на събитие, предоставяйки име, описание, дата/час, цена(в случай на платено събитие), организация(в случай на принадлежност на събитието към организация)
3. Създаване на въпроси и отговори за дадено събитие
4. Генериране на линк, служещ като покана за събитието, чрез който единствено регистриран портебител има право на достъп
5. Отговаряне на въпроси за събитието от регистриран потребител
6. Извършване на плащане - транзакция чрез банкова карта, в случай на платено събитие
7. Генериране на статистики на база отговорите на въпросите за дадено събитие
8. Създаване на организация
9. Преглед на детайлите на събитие
10. Създаване на срещи - представляват чат, намиращ се на страницата с детайлите за събитието. Служат за дискусия между участниците.
11. Проверка на брой плащания, извършени за събитие
12. Изпращане на покана по електронна поща

# 2. Основни технологии, използвани за изготвяне на приложението

## React

Проектът използва един от най-използваните фрейуърци за изграждане на Single Page Application-и. Бърз, надежден и удобен за структуриране на големи приложения, използва виртуален DOM за следене за промени на DOM дървото. Рендерирането на компонентите се извършва на клиентската част.

## Material-UI

Дизайн системат, намаляваща времето за стилизация, като същевременнопредоставя функционални и добре стилизирани компоненти с матириъл дизайн

## Node JS

За изграждане на сървърната(отдалечената) част на приложението е използван един от най-популярните средa за изпълнение на сървърни и мрежови приложения с отворен код на JavaScript. Oптимизира производителността и мащабируемостта на дадено приложение.Съдържа вградени библиотеки, които позволяват приложенията да работят като сървъри, без софтуер като Apache HTTP Server или IIS.

**Express JS**

Фреймуъркът е предназначен за изграждането на уеб приложения и различни API-а. Позволяваудобна обработка на http заявките, както и лесна дефиниция на сървърните пътищата на приложението. В проекта се използва за направата на REST API частта.

## 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.

**Indicative**

Библиотека, използвана за сървърна валидация на входните дании. В проекта се използва се преди създаването на моделите за валидация на необходимите полета.

**React router dom**

Библиотека, служеща за навигация между отделните компоненти в проекта.

**nodemailer**

Лесна и удобна библиотека за изпращане на мейли. В проекта се използва сървис на gmail

**Socket.IO**

Библиотек на JavaScript за уеб приложения в реално време. Тя дава възможност в реално време, двупосочна комуникация между уеб клиенти и сървъри. В проекта е използвана за направата на чат между участниците в събитие.

# 3. По-важни спецификации на проекта

Приложението използва Single Page архитектурата за изграждане на front-end частта, като по този начин се предоставя по-бързо и динамично навигиране между страниците, без необходимост от презареждане при всяка една от тах

Сървърната част е реализирана чрез REST дизайн, използвайки NodeJs/ExpressJs като сървърни среди и MongoDB като база от данни.

За удобство е излозван Mongoose ODM.

Следва описание на заявките към отдалечената, предоставяща информация на визуалната, част на приложението:

### Потребители

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.

### Събития

Нов събитие може да бъде създадено чрез POST заявка към сървъра.

При създаване, ако са изпратени въпроси в заявката, автоматично се запазват заедно с съответните им отговори, отново част от заявката.

Преглед на всички събития се осъществява с GET заявка.

### Срещи

Нов среща може да бъде създадена чрез POST заявка към сървъра.

Тя е уникална в рамките на събитието, като се връща грешка при вече съществуваща такава.

**Организации**

Нов организация може да бъде създадена чрез POST заявка към сървъра.

Системата предоставя възможност за извличане на организации, за които съзтател е потребителя, както и събития, принадлежащи на дадена организация – и двете чрез GET заявки.

**Плащания**

Плащанията се създават с POST заявкa, като се запазва референция към събитието, цената и референция към потребителя, извършил плащането.

**Въпроси към събитията**

Въпросите се запазват при създаване на дадено събитие с POST заявка. Всеки въпрос принадлежи към едно събитие

**Отговори към събитията**

Възможните отговори се запазват при създаване на дадено събитие с POST заявка. Всеки отговор принадлежи към един впрос.

### Организация на информацията - релации м/у отделните домейни

Всяко **събитие** е отделна независима релация, състояща се от полетата заглавие, описание, дата, час, цена. Съществува релация между него и релацията на въпросите, като така към всяко събитие могат да бъдат дефинирани множество **въпроси**. Въпросите се запазват като масив от идентификатори.

Всеки **потребител** е независима релация, състояща се от имейл, хеширана парола, първо име, второ име .

Всеки **въпрос** е релация, която има връзка към релацията на събитията тип едно към много. По този начин много въпроси могат да бъдат задавани към дадено събитие. Релацията се състои от съдържание – описание на въпроса и масив от идентивикатори към **отговори** – друг тип релация, съхраняваща възможните отговори за конкретния въпрос.

Всяка **организация** се състои от заглавие, референция към създателя й от таблицта на потребителите и описание.

Всяка **среща** е релация, принадлежаща към едно единствено събитие. Състои се от заглавие, описание референция към събитието и референция към създателя на потребителя.

Всяки **съобщение** е релация, принадлежаща към единствена среща. Съдържа референция към потребителя, изпратил съобщението, съдържанието на съобщението и референция към срещата, към която принадлежи.

Всяко **плащане** е релация, състояща се от референция към събитието, за която се отнася, референция към потребителя, извършил плащането и цена.

Съществъва **мапинг таблица**, запазваща отговорите на въпросите на различните потребители.

# 4. Линк към source code-а на приложението

[https://github.com/torqux1/event-management-system](https://github.com/alien1976/shopping-assistant-react)

# 

# 5. Използвани материали за изготване на приложението

* React - <https://reactjs.org/>
* ExpressJS - <https://expressjs.com/>
* MongoDB - <https://www.mongodb.com/>
* MaterialUI - <https://material-ui.com/>
* Indicative - <https://indicative.adonisjs.com/>
* nodemailer - <https://nodemailer.com/about/>
* Socket.IO - <https://socket.io/>