

КУРСОВ ПРОЕКТ

Дисциплина: ПРОЕКТИРАНЕ И ИНТЕГРИРАНЕ НА СОФТУЕРНИ СИСТЕМИ

РЕАЛИЗАЦИЯ НА СИСТЕМАТА

Версия 1.x

Фак. №	ИМЕ НА СТУДЕНТ	СЕКЦИЯ ОТ ДОКУМЕНТА
ОМІ0600124	Борил Диянов Игнатов	1,2,3,4,5,6 и 7

Януари, 2025

Съдържание

1	Въведение	3
1.1	Цел	3
1.2	Резюме.....	3
1.3	Дефиниции и акроними	3
2	Използвани технологии	5
3	Реализация на базата от данни	6
4	Реализация на бизнес логиката.....	7
5	Реализация на потребителския интерфейс.....	8
6	Внедряване на системата.....	11
7	Разпределение на дейностите по реализацията	12
8	Приложения	13

1 ВЪВЕДЕНИЕ

1.1 Цел

Този документ представя внедряването на системата EventSphere, предназначена да предоставя информация за различни видове събития, включително културни, спортни и други.

Основната цел на софтуера е да улесни потребителите при избора на подходящо и привлекателно за тях събитие, като взема под внимание техните интереси, финансови възможности и местоположение.

1.2 Резюме

Основната задача на системата е да открива предстоящи събития в съответствие с предпочитанията на потребителите, което изисква обработка на всички налични събития. За постигането на тази цел системата използва принципите на разпределени софтуерни архитектури, които осигуряват бърза обработка на заявките. Композиционните модули са самостоятелни и независими, като по този начин се избягва конкуренцията между задачите.

Системата се състои от три основни модула:

- **REST API** – Основният компонент на системата, базиран на REST стандарта, който гарантира stateless архитектура. За съхранение на потребителски данни се използва база данни. Основната функционалност на този модул е да открива събития, които отговарят на изискванията на потребителите. Заявките от потребителския интерфейс се изпращат директно към този компонент, като се филтрират на база нивото на достъп на съответния потребител чрез JWT токени.
- **Потребителски интерфейс** – Този модул предоставя визуализация на информацията, генерирана от REST API. Потребителите могат да видят информация за предстоящи и минали събития, както и данни за други потребители на системата.
- **Система за анализ и филтриране на потребителски изображения** – Интерфейсът позволява на потребителите да качват изображения за профилите си, които са видими за останалите. За да отговарят на определени критерии за съдържание, тези изображения преминават през филтрационния компонент на системата, който оценява дали съответстват на изискванията.

1.3 Дефиниции и акроними

- **Rest API** – Уеб интерфейс, който работи според принципите на HTTP и REST, за да осигури комуникация между различни софтуерни компоненти.
- **Потребителски интерфейс** – Визуален интерфейс, предоставящ на потребителя елементи за управление, представени като графични изображения (например менюта, бутони, списъци и други).
- **Разпределена софтуерна архитектура** – Архитектурен модел, при който всеки модул е самостоятелна и независима единица.
- **Приложение (app, application)** – Софтуер, създаден с цел да помогне на потребителя при изпълнение на конкретна задача.
- **Модул** – Логически разделена софтуерна единица, която изпълнява определена функционалност.

- **Уеб приложение** – Приложение, достъпно за потребителите чрез мрежа, като например Интернет.
- **Потребител** – Лице, което използва компютърна система или мрежова услуга.
- **База от данни** – Организирана колекция от информация, създадена за лесно достъпване, управление и актуализиране.
- **Сървър** – Активна инстанция на софтуер, която приема заявки от клиенти и връща подходящи отговори.
- **Клиент** – Компонент на компютърна или софтуерна система, който използва услуга, предоставяна от сървър.

2 ИЗПОЛЗВАНИ ТЕХНОЛОГИИ

За хостинг на сървъра използвам VM на AWS платформата.

Избраната операционна система е Amazon Linux.

Използва се Java spring boot.

За база данни се използва postgresSQL.

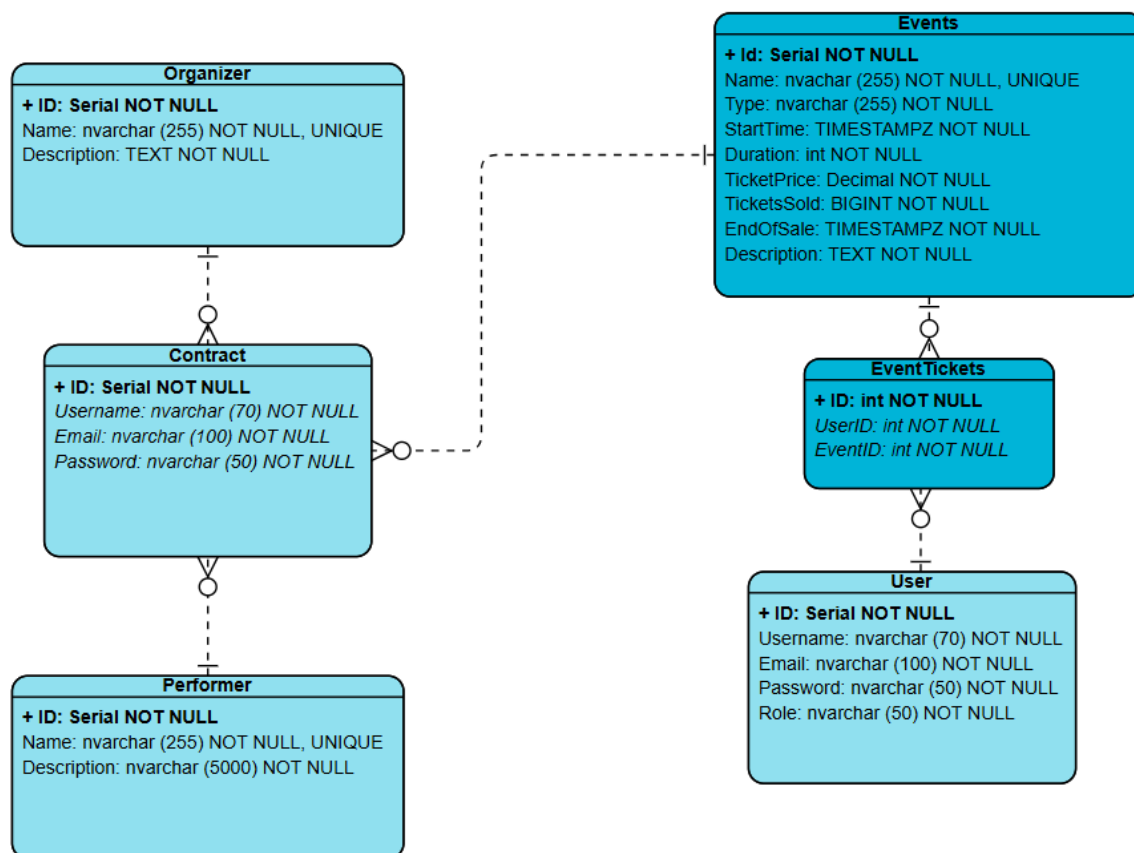
Junit и Mockito за компонентно тестване на back-end частта.

За реализацията на потребителския интерфейс са използвани HTML, CSS и Javascript.

Интегриран е python сървър, който комуникира чрез сокет със spring boot.

Използва се embedded tomcat сървър, тъй като добре работи с java spring boot.

3 РЕАЛИЗАЦИЯ НА БАЗАТА ОТ ДАННИ



Схемата на същности отношения, описана в документацията за проектирането на системата, налага използването на връзката един към много. Таблицата „Contract“ съдържа всички връзки между организатори, изпълнител и събитие.

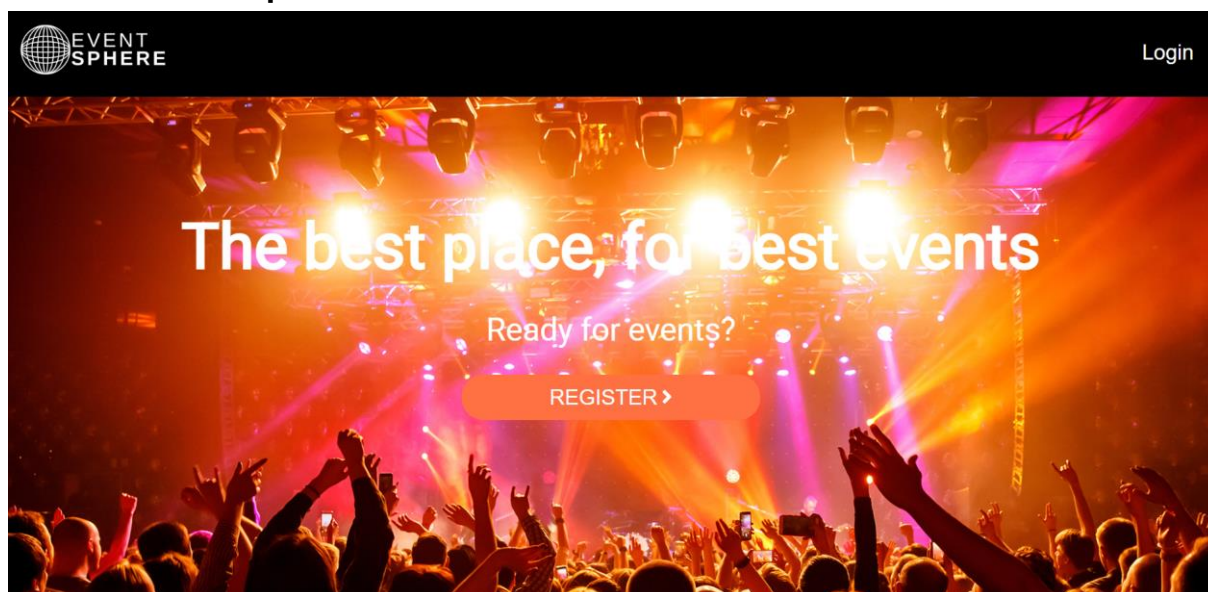
4 РЕАЛИЗАЦИЯ НА БИЗНЕС ЛОГИКАТА

Бизнес логиката е разработена с помощта на Java Spring Boot framework, който позволява ефективна и модулна организация на сървърната част. Сървърната архитектура е разделена на отделни подмодули, наречени Manager-и (например: User Manager, Event Manager, File Upload Manager). Всеки Manager е допълнително структуриран в няколко логически единици, всяка със своята конкретна роля в рамките на Spring Boot приложението:

- **Model:** Представява основните обекти в системата, като например User, Event и други, заедно с техните атрибути. За тях се дефинират конструктори, както и setter и getter методи. Всеки Model обикновено съответства на таблица в базата данни, използвайки Object Relational Mapping (ORM) с Hibernate ORM, което улеснява управлението на базата данни.
- **Repository:** Тези компоненти осъществяват комуникацията между бизнес логиката и базата данни. Чрез ORM автоматично се създават основните SQL операции (CRUD: Create, Read, Update, Delete), като се премахва нуждата от ръчно писане на SQL заявки. Това прави приложението независимо от типа база данни и нейния SQL синтаксис. Конфигурацията се извършва само чрез свързване на Spring Boot приложението към избраната база данни.
- **Service:** В този слой се реализира същинската бизнес логика. Тук се обработва взаимодействието между различните обекти и се осигурява изпълнението на функционалните изисквания на системата.
- **Controller:** Свързва HTTP заявките, изпратени от клиентската част (например браузър), с подходящите endpoint-и на сървърната страна. Получените данни се предават към съответния Service за обработка, след което отговорът се връща на клиента във вид на HTML страница, JSON формат или съобщение за грешка.
- **DTO (Data Transfer Object):** Служи за пренос на данни между различни компоненти, по-конкретно между клиентската и сървърната част. DTO се преобразува в Model чрез Assembler-и, за да се обработи в Service слоя, и обратно – Model в DTO, за да се върнат данни във формат, подходящ за front-end частта.

User Manager модулът е изграден с помощта на Spring Security, специализиран модул на Spring framework за управление на сигурността. Освен традиционните функции като регистрация, вход и изход от системата, е внедрена и JWT (JSON Web Tokens) автентикация. Това осигурява по-високо ниво на защита и по-сигурна комуникация между потребителите и системата.

5 РЕАЛИЗАЦИЯ НА ПОТРЕБИТЕЛСКИЯ ИНТЕРФЕЙС



Фиг. 1 - Начален екран

Register

Username

Email

Password

Confirm Password

Credit card details

Visa

Name on card

Card number

Expiry date

CVV code

Register

Фиг. 2 - Форма за регистрация

Login

Username








Password

Continue

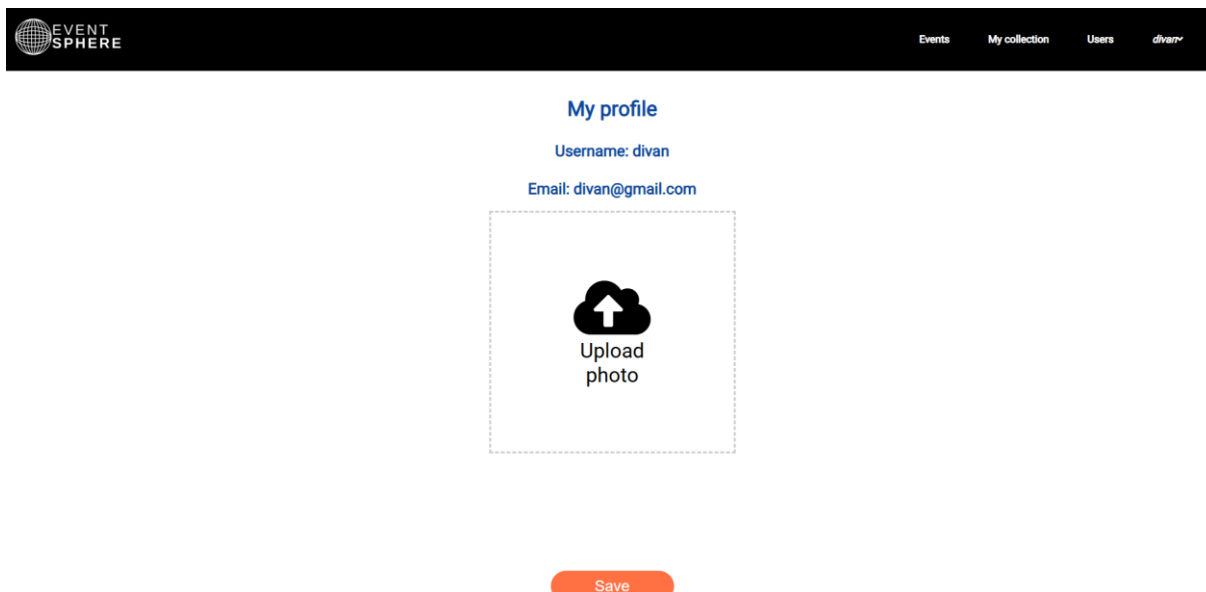
Not a member? Sign up!

Фиг. 3 – Форма за вход

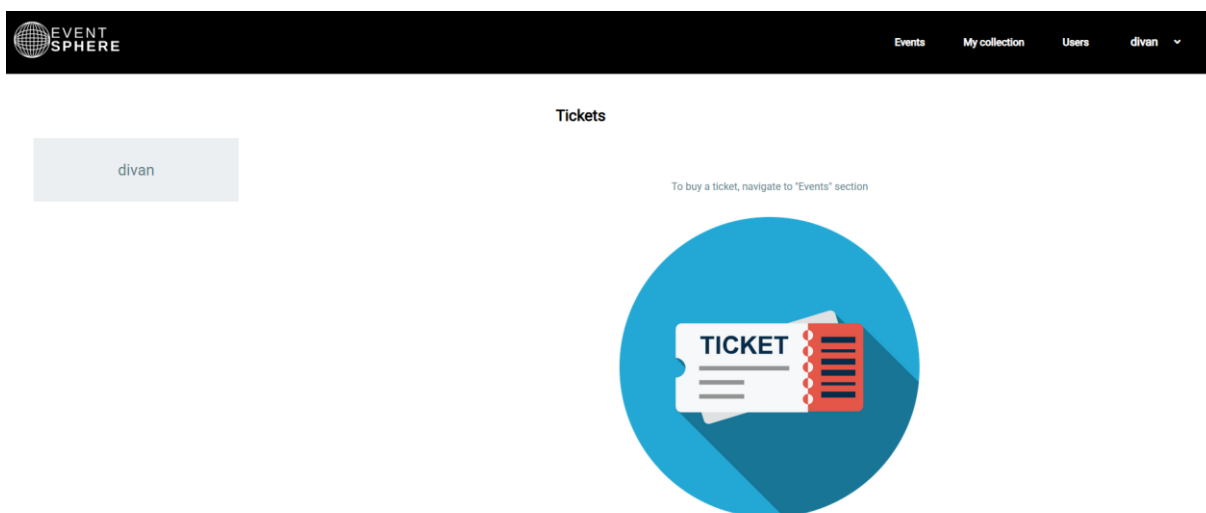
Users

 borko b.d.ignatov@gmail.com	 ivan ivan@gmail.com	 stefan stefan@abv.bg	 johndoe johndoe@abv.bg
 donald donald@abv.bg	 park park@china.bg	 divan divan@gmail.com	

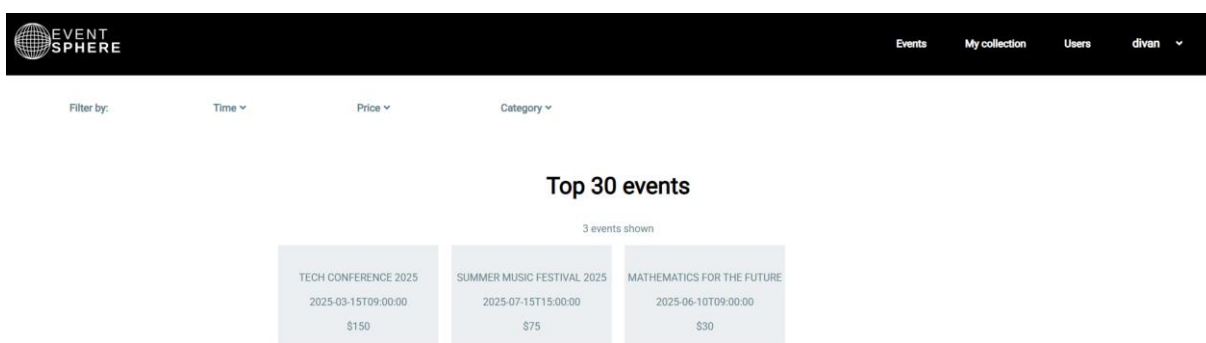
Фиг. 4 – Списък с потребителите



Фиг. 5 – Моят профил



Фиг. 6 – Моята колекция от събития



Фиг. 7 – Всички събития

6 ВНЕДРЯВАНЕ НА СИСТЕМАТА

1. Резервирах си инстанция за виртуална машина на AWS.
2. След това инсталирах Java 17.
3. Добавих Maven и Spring boot.
4. Добавих PostgreSQL база данни за spring сървъра.

7 РАЗПРЕДЕЛЕНИЕ НА ДЕЙНОСТИТЕ ПО РЕАЛИЗАЦИЯТА

Борил Игнатов изпълни цялостната реализация на системата, като се погрижи за разработката на архитектурата от бази данни, front-end и back-end разработката, системната администрация.

8 ПРИЛОЖЕНИЯ

Тази секция не е задължителна и се използва при необходимост.

Приложенията се включват директно или се реферират.