Федеральное государственной образовательное учреждение
Высшего профессионального образования
Национальный исследовательский технологический университет «МИСиС»
Институт информационных технологий и компьютерных наук (ИТКН)

Курсовая работа по курсу «Программирование клиент-серверных приложений»

Тема: «Форум»

Выполнила: студентка 2 курса, БИВТ-21-6
Герасимова Мария Сергеевна
Руководитель: Рзазаде У.А.

Москва 2023

Предметная область

Работа по созданию клиент-серверного приложения является крайне актуальной в настоящее время и остается востребованной в ближайшей перспективе. Созданное на основе клиент-серверной архитектуры, позволяет легко масштабироваться и управлять большим объемом информации. Это также обеспечивает высокую производительность, что является ключевым фактором во время выбора приложения пользователем.

Такое приложение может быть использовано в различных сферах бизнеса и сферы развлечения: от блогов по увлечениям до серверов больших бизнес-предприятий. Разработка веб-приложений в целом является стратегически важной для любой компании, которая хочет оставаться конкурентоспособной в современном мире бизнеса.

Постановка задачи

Необходимо разработать клиент-серверное приложение – форум по теме №22 (Форум, пользователь, посты, комментарии). Клиентское приложение должно предоставлять пользователям удобный и интуитивно понятный интерфейс. Серверное приложение должно обрабатывать запросы от клиентов, хранить информацию о клиентах и блогах.

Основные функциональные требования к приложению:
- Поиск блога, пользователя и других страниц по имени или его аналогу или айди;
- Возможность создания нового блога или изменение старого, удаление блога;
- Возможность создания, удаления, редактирования комментариев;
- Регистрация, удаление и изменение профилей пользователей;

Для написания данной работы были выбраны следующие материалы: TypeScript для написания серверной часть, MySql для создания базы данных, HTML для создания клиентской части проекта.
В качестве тематики форума был выбран форум по увлечениям. Форум содержит страницы-блоги с информацией о музыкальных группах и их исполнителях, а также обсуждение, прикрепленное к этой группе, осуществляемое комментариями пользователей.

Описание архитектуры

В курсовой работе используется простейшая трехслойная архитектура клиент-сервисного приложения:

Клиент <-> Сервис <-> База Данных.

1. Клиентская часть - это приложение, установленное на устройстве пользователя и предоставляющее интерфейс для взаимодействия с сервисом. Она может быть реализована с помощью HTML/CSS/JavaScript или любого другого фреймворка, в данной работе клиентская часть реализована на HTML. В клиентской части реализуется интерфейс для работы с корзиной, браузером товаров и авторизации пользователей.
2. Серверная часть - это приложение, установленное на сервере, которое обрабатывает запросы от клиентской части, взаимодействуя с базой данных. Серверная часть может быть написана с использованием стека технологий (например, Node.js, Python, Ruby on Rails). В серверной части должны быть реализованы скрипты для обработки запросов на создание, удаление и редактирование профилей, блогов и комментариев.
3. База данных - это хранилище информации о пользователях, блогах, комментариях и других данных, необходимых для работы приложения. Для реализации базы данных обычно используется одна из современных реляционных баз данных, например, MySQL, PostgreSQL, SQLite.

Взаимодействие между слоями происходит посредством протокола HTTP, где клиент отправляет запрос на сервер, сервер обрабатывает его и отправляет ответ клиенту. Реализованная таким образом архитектура позволяет обеспечить масштабируемость системы, легкость совместной работы множества разработчиков и быструю обработку запросов от пользователей.

Описание структуры БД

4.1 Общее описание

Исходя из требований к системе была разработана схема базы данных, представленная на рисунке .

Также на этапе проектирования были описаны названия всех таблиц и полей базы данных.

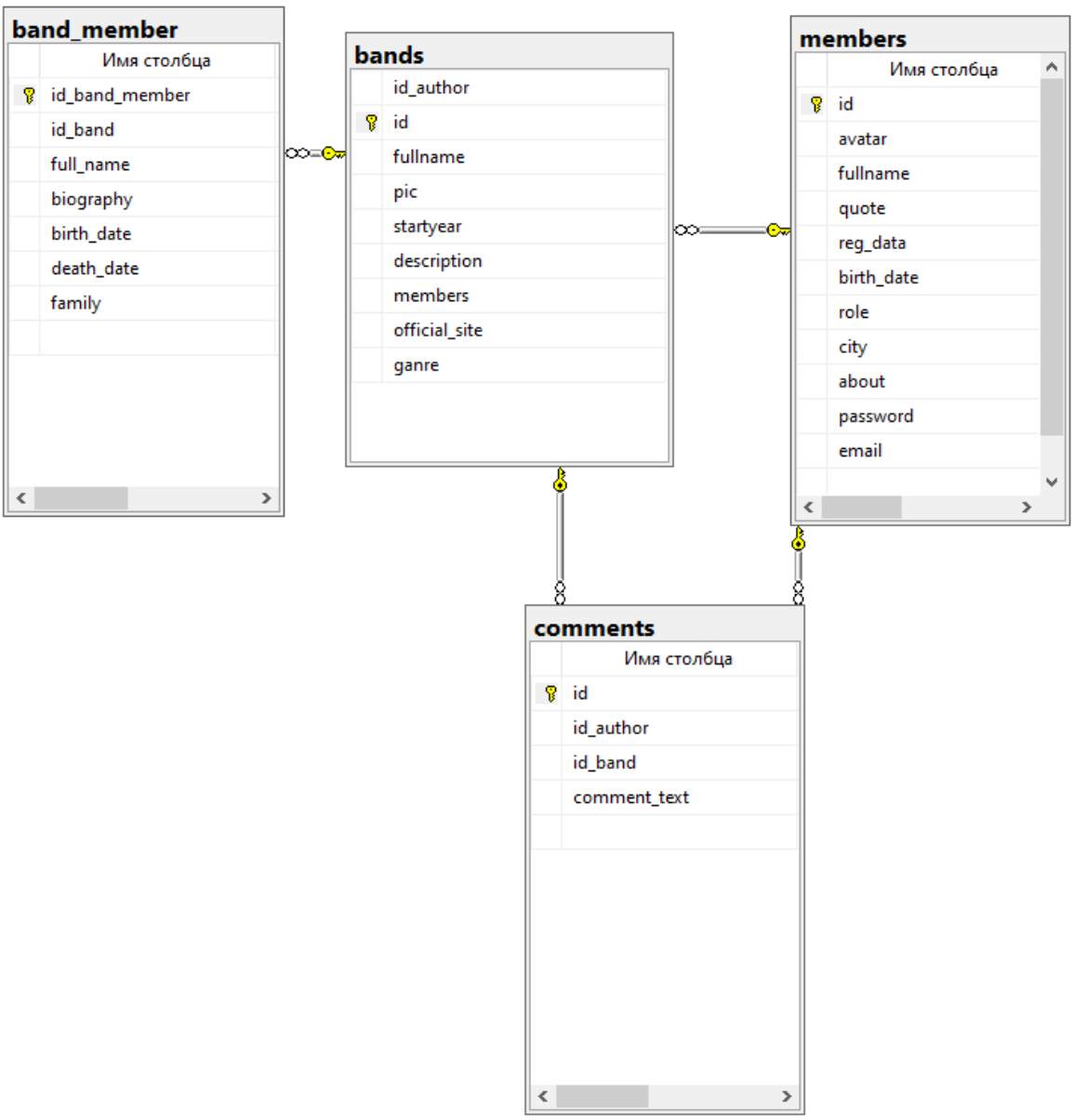

Рис.1 Структура базы данных

4.2 Таблица Members

Таблица members: имеет данные о пользователе сайта.

ID – уникальный номер пользователя, выдаваемый автоматически при создании аккаунта. Принимает цифровые значения. Поле не может быть пустым.

Avatar – аватар пользователя. Принимает изображение как значение. Поле может быть пустым.

Fullname – никнейм пользователя. Принимает строку как значение. Поле не может быть пустым.

Quote – статус пользователя, любая строка, заполняемая самим пользователем. Принимает строку как значение. Поле может быть пустым.

Reg_data – дата регистрации, записывающаяся автоматически при создании аккаунта. Принимает дату как значение. Поле не может быть пустым.

Birth_date – дата рождения пользователя, выставляется пользователем по желанию. Принимает дату как значение. Поле может быть пустым.

Role – роль пользователя на сайте (Пользователь, Модератор, Администратор), первый статус выдается автоматически. Принимает строку как значение. Поле не может быть пустым.

City – город пользователя, заполняется по желанию. Принимает строку как значение. Поле может быть пустым.

About – основная информация о пользователе в произвольном виде. Принимает строку как значение. Поле может быть пустым.

Password – пароль пользователя. Принимает строку как значение. Поле не может быть пустым.

Email – электронная почта пользователя. Принимает строку как значение. Поле не может быть пустым.

4.3 Таблица Bands

Таблица bands: содержит информацию о страницах музыкальных групп с сайта.

Id_author – ID пользователя-модератора страницы о группе. Принимает числовое значение из группы Members.

Id- уникальный номер страницы, выдаваемый автоматически. Принимает числовое значение. Поле не может быть пустым.

fullname – название группы. Принимает строку как значение. Поле не может быть пустым.

pic – главная фотография на странице группы. Принимает изображение как значение. Поле может быть пустым.

startyear – дата начала карьеры группы. Принимает строку как значение. Поле может быть пустым.

description – описание группы в свободном виде. Принимает строку как значение. Поле может быть пустым.

members – участники группы. Принимает строку как значение. Поле может быть пустым.

official_site – ссылка на официальный сайт группы. Принимает строку как значение. Поле может быть пустым.

ganre – жанры группы. Принимает строку как значение. Поле может быть пустым.

4.4 Таблица Comments

Таблица comments: содержит информацию о комментариях под блогами.

Id - уникальный номер комментария, выдаваемый автоматически. Принимает числовое значение. Поле не может быть пустым.

Id_author - ID пользователя, который оставил комментарий. Принимает числовое значение из группы Members.

Id_band - ID страницы группы, под блогом которой оставили комментарий. Принимает числовое значение из группы Bands.

Comment_text – основное содержание текста комментария. Принимает строку как значение. Поле не может быть пустым.

4.5 Таблица Band_member

Таблица band_member: содержит информацию со страниц об участниках групп.

Id_band_member – уникальный номер пользователя, выдаваемый автоматически при создании страницы. Принимает цифровые значения. Поле не может быть пустым.

Id_band – ID страницы группы, к которой приписан участник. Принимает числовое значение из группы Bands.

Full_name – имя участника. Принимает строку как значение. Поле не может быть пустым.

Biography – биография участника, изложенная в свободной форме. Принимает строку как значение. Поле может быть пустым.

Birth_date – дата рождения исполнителя. Принимает дату как значение. Поле может быть пустым.

Death_date – дата смерти исполнителя. Принимает дату как значение. Поле может быть пустым.

Family – информация о семейном положении участника. Принимает строку как значение. Поле может быть пустым.

Описание серверной части

5.1 Сущности

Для выполнения работы приложения было создано 4 сущности, все классы являются объектами типа TypeORM:

Band_member – экспортирует класс Band_Member, имеющий 6 свойства: id, fullname, biography, birth_date, death_date, family. Описывает JOIN-таблицы с классом affiliation и Band.

Band – экспортирует класс Band, имеющий 8 свойств: id, fullname, pic, startyear, description, members, ofsite, ganre. Описывает JOIN-таблицы с классом affiliation и Band_Member.

Comment – экспортирует класс Comment, имеющий 4 свойства: id, senderID, forumname, text. Описывает JOIN-таблицы с классом affiliation и Member.

Member – экспортирует класс Comment, имеющий 10 свойств: id, fullname, quote, red_data, bir_data, role, city, about, password, email. Класс имеет API-декораторы, предоставляющие примеры и описание каждого свойства. Описывает JOIN-таблицы с классом affiliation и Comment.

Пример созданной сущности:

```
@Entity('comments')
export declare class Comment {
    @PrimaryGeneratedColumn()
    id: number;
    @Column()
    senderID: number;
    @Column()
    forumname: string;
    @Column()
    text: string;
    @ManyToMany((type) => Member, (member) => member.comments)
    @JoinTable({
      name: 'comment_affiliation',
      joinColumn: { name: 'comment_id' },
      inverseJoinColumn: { name: 'affiliation_id' },
    })
    members: Member[];
    @ManyToMany((type) => Comment_Affiliation, (affiliation) =>
affiliation.comments)
    @JoinTable({
      name: 'comment_affiliation',
      joinColumn: { name: 'comment_id' },
```

```
      inverseJoinColumn: { name: 'affiliation_id' },
    })
    affiliations: Comment_Affiliation[];
  }
```

5.2 Классы аффилиации

Классы affiliation – были созданы для поддержания связей между классами. Для каждого класса был создан свой affiliation класс из-за существования множества связей. Классы имеют аналогичную информацию с классами, для которых они были созданы.

Пример на классе Band_Member_Affiliation:

```
       @Entity('affiliations')
     export class Band_Member_Affiliation {
         @PrimaryGeneratedColumn()
         id: number;
         @Column()
         fullname: string;
         @Column()
         biography: string;
         @Column()
         birth_data: string;
         @Column()
         death_data: string;
         @Column()
         family: string;
       @ManyToMany((type) => Band_Member, (band_member) =>
     band_member.affiliations)
       @JoinTable({
         name: 'band_member_affiliations',
         joinColumn: { name: 'affiliations_id' },
         inverseJoinColumn: { name: 'band_member_id' },
       })
       band_members: Band_Member[];
}
```

5.3 Configuartion

Configuration – класс созданный для подключения к базе данных. Имеет информацию для верификации подключения.

Класс configuration:

```
const ormConfig: DataSource = new DataSource({
  type: 'mysql',
  host: 'localhost',
  port: 5432,
  database: 'education',
  username: 'education',
  password: 'password',
  entities: ['dist/**/*.entity{.ts,.js}'],
  logging: true,
  synchronize: false,
  migrationsTableName: 'migrations',
  migrations: ['dist/src/migrations/*{.ts,.js}'],
});
export default ormConfig;
```

5.4 Контроллеры

Контроллеры – классы, созданные для обращения на сервер. В методах класса происходит обработка клиентского запроса. Контроллеры были созданы для каждой сущности и классов DTO соответственно. Каждый контроллер имеет функцию:

findAll -нахождение всех записей соответствующего типа. Не принимает аргументов. Get-запрос.

findOne – нахождение одной записи по ее ID номеру. Принимает число как аргумент – ID который находится. Get-запрос.

Update – функция для обновления записи. Как аргумент принимается ID записи и класс обновляемого объекта. Put-запрос.

Create – функция для создания записи. Как аргумент принимается ID класс обновляемого объекта. Post-запрос.

Remove – функция удаления записи. Принимает число как аргумент – ID, запись которого должна быть удалена. Delete-запрос.

Пример класса контроллера:

```
@Controller('authors')
@ApiTags('Пользователи')
export class MembersController {
```

```
    constructor(private readonly membersService:
  MembersService) {}

  @Get()
    findAll() {
      return this.membersService.findAll();
    }

  @Get(':id')
    findOne(@Param('id') id: string) {
      return this.membersService.findOne(+id);
    }

  @Put(':id')
    update(@Param('id') id: string, @Body() updateMember:
  Member) {
      return this.membersService.update(+id, updateMember);
    }

  @Post()
    create(@Body() createMember: CreateMemberDto) {
      return this.membersService.create(createMember);
    }

    @Delete(':id')
    remove(@Param('id') id: string) {
      return this.membersService.remove(+id);
    }
}
```

5.5 Datasourse

Datasource классы созданы для модуля и сервиса соответственно. Созданная ранее псевдобаза данных для содержания источника данных. Содержит списки классов.

5.6 DTO-объекты

DTO-объекты созданы для разделения сущностей, хранящиеся в базе данных и передаваемые пользователю. На каждую сущность был создан свой DTO-объект, хранящий все свойства класса.

Пример DTO-объекта:

```
export class CreateBandDto {
    fullname: string;
    pic: string;
    startyear: number;
    description: string;
    members: string;
    ofsite: string;
    ganre: string;
    affiliations: number[];
}
```

5.7 Модули

Модули сущностей созданы для управления всеми компонентами в приделах своих сущностей, включая их контроллер и сервис. Модуль определяет контроллер, сервис, импортируемый класс и TypeORM модели.

Пример модуля:

```
@Module({
  controllers: [BandsController],
  providers: [BandsService],
  imports: [DatasourceModule,
  TypeOrmModule.forFeature([Band, Band_Affiliation]), ]

})
export class BandsModule {}
```

5.8 Сервисы

Сервисы сущностей являются обработчиками высоконагруженных операций. Сюда входит асинхронная логика взаимодействия с базой данных.

В сервисах описана работа создания репозиториев и некоторые запросы для обработки записей базы данных.

Пример сервиса:

```
@Injectable()
export class CommentsService {
 constructor(
  @InjectRepository(Comment)
  private readonly commentRepository: Repository<Comment>,
 ) {}
 async create(commentDto: CreateCommentDto): Promise<Comment>
 {
  const comment = this.commentRepository.create();
  comment.senderID = commentDto.senderID;
  comment.forumname = commentDto.forumname;
  comment.text = commentDto.text;
  await this.commentRepository.save(comment);
  return comment;
 }
 findOne(id: number): Promise<Comment> {
  return this.commentRepository.findOne({
   where: { id },
  });
 }

 async findAll(): Promise<Comment[]> {
  const comments = await this.commentRepository.find({
  });
  return comments;
 }

 remove(id: number) {
  this.commentRepository.delete({ id });
 }
}
```

5.9 Главный код

Главный код создает путь для подключения сервиса и его дальнейшего тестирования:

```
async function bootstrap() {
```

```
  const app = await NestFactory.create(AppModule);
  const config = new DocumentBuilder()
    .setTitle('Education API')
    .setVersion('1.0')
    .build();
  const document = SwaggerModule.createDocument(app, config);
  SwaggerModule.setup('api_docs', app, document);
  await app.listen(3001);
  await app.setGlobalPrefix('/api');
 }
bootstrap();
```

Финальная архитектура проекта выглядит следующим образом:

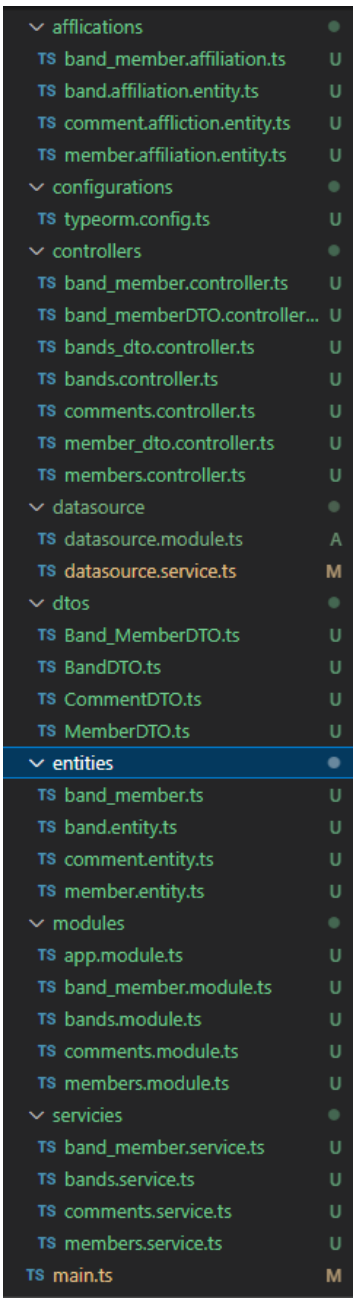

Рис.2 Архитектура проекта

Описание клиентской части

Клиентская часть создана с помощью HTML языка и представляет собой объединенную группу WEB-страниц.

6.1 Главная страница

Главной страницей форума является приветственная страница с краткой информацией о сайте в середине страницы. Вверху страницы существует меню навигации по главным страницам сайта (1), которое сохраняется на всех страницах сайта и является закрепленным

элементом. Для незарегистрированных пользователей существует ссылка на форму создания профиля (2). Внизу выводятся случайные блоги групп (3), каждое изображение имеет ссылку на переход к обсуждению.

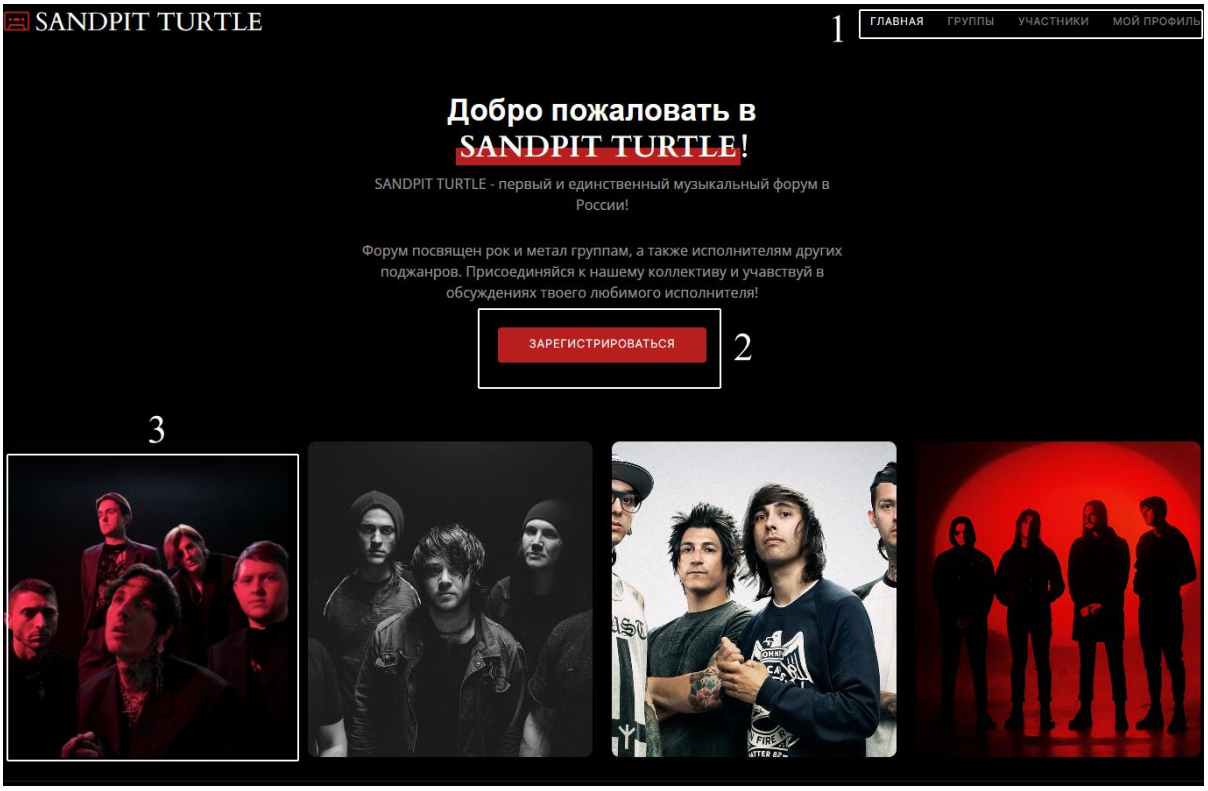

Рис.3.1 Клиентский интерфейс: главная страница

6.2 Вкладка «Группы»

Во вкладке «Группы» имеется полный список обсуждений. Пользователь может создать собственное обсуждение с помощью кнопки (1). Сохраняется навигационное меню. Каждое изображение ведет в обсуждение соответствующей группы.

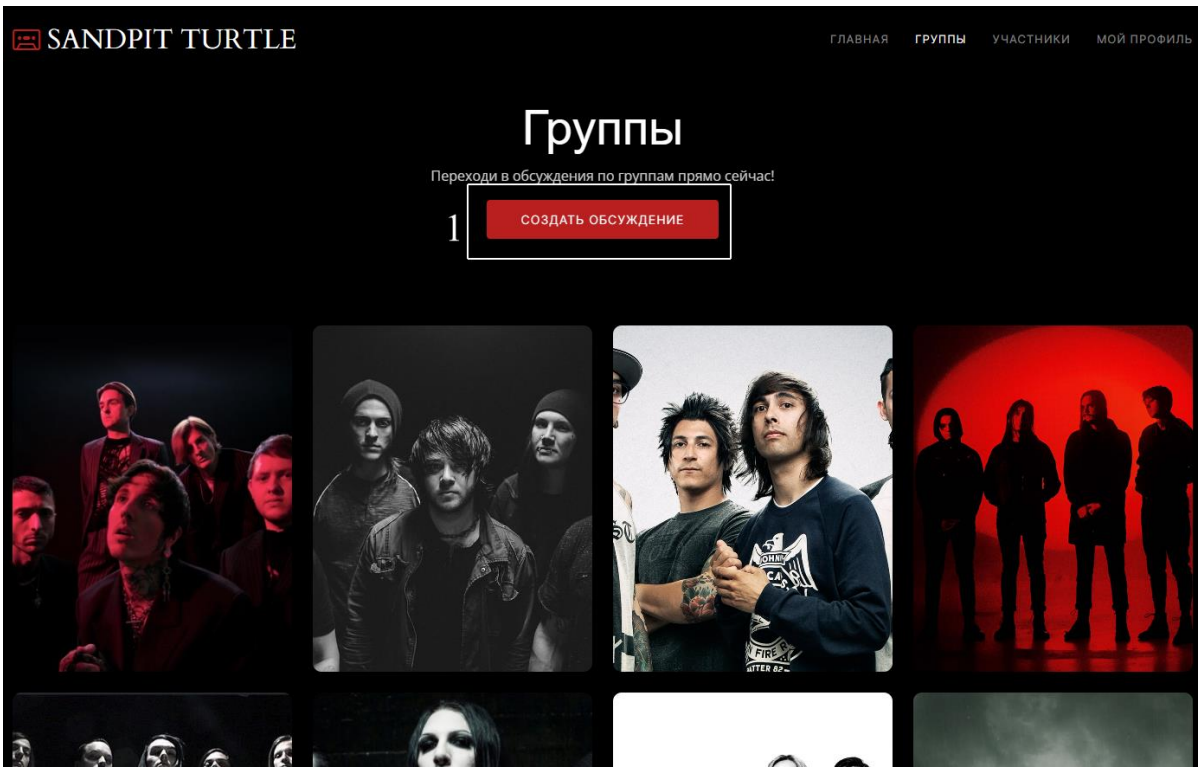

Рис.3.2 Клиентский интерфейс: вкладка «Группы»

6.3 Вкладка «Участники»

Вкладка «Участник» предоставляет полный список пользователей сайта. Для незарегистрированных пользователей существует ссылка на форму создания профиля (1). Для каждого участника отображается: его аватар (2), никнейм (3) и некое «О Себе» - описание участника в свободной форме. При нажатии на аватарку или имя пользователя переносит на страницу участника.

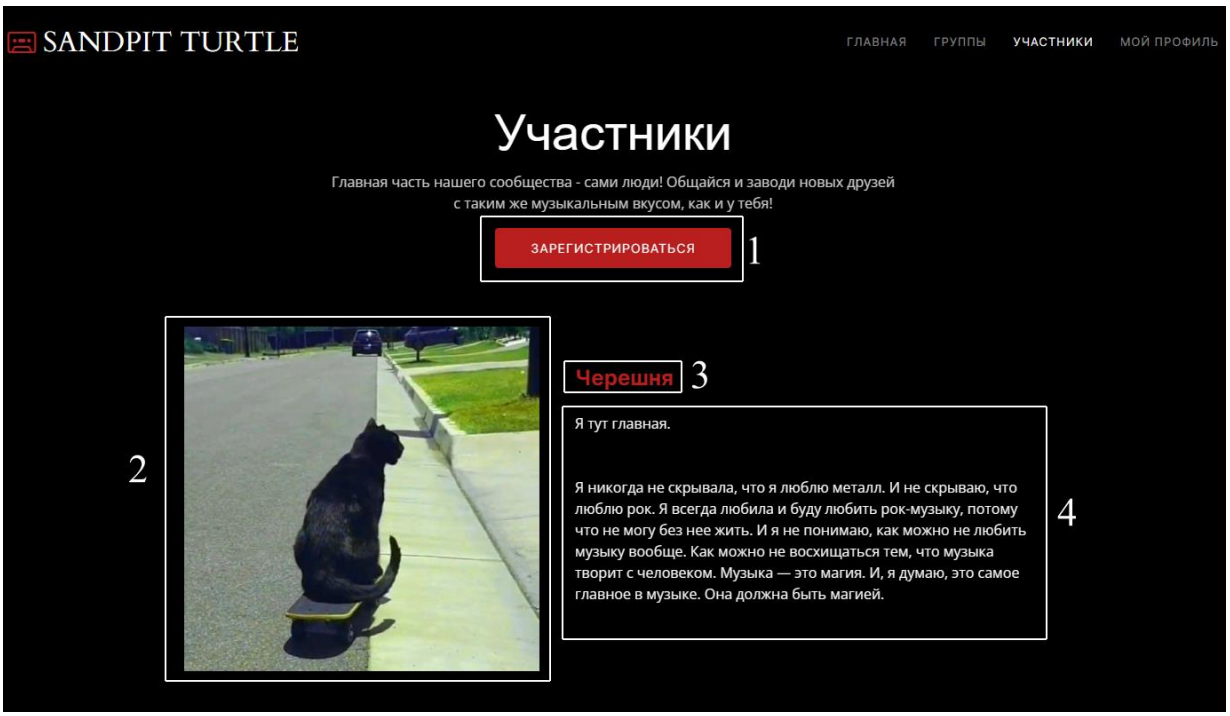

Рис.3.3 Клиентский интерфейс: вкладка «Участник»

6.4 Профиль пользователя

Профили участников и собственный профиль отображаются одинаково. На странице размещена вся информация об участнике: его аватарка, никнейм, статус, информация, заполненная по желанию участника, роль на сайте, администрируемые обсуждения.

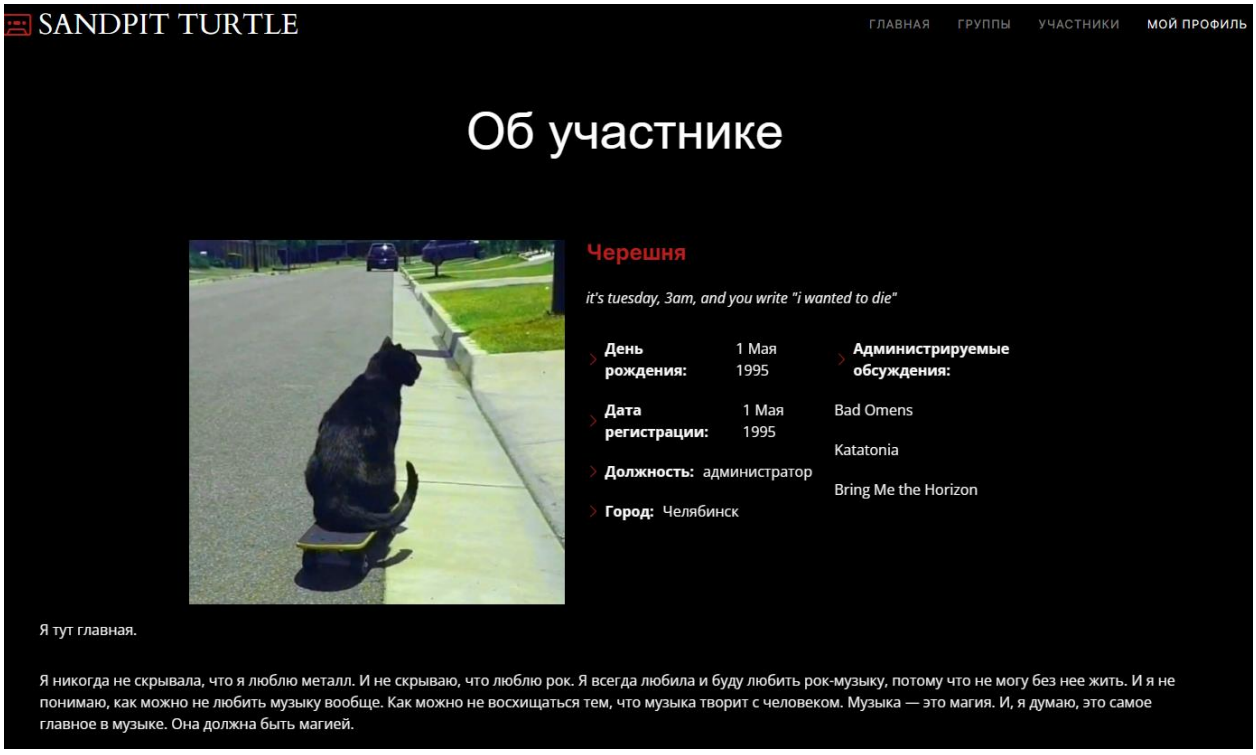

Рис.3.4 Клиентский интерфейс: профиль пользователя

6.5 Страница обсуждения

В обсуждениях на страницах группы содержится информация о самом музыкальном составе: отображается главная фотография для группы, описание, краткая сводка.

Обсуждение представлено в комментариях, размещенных снизу основного блока темы.

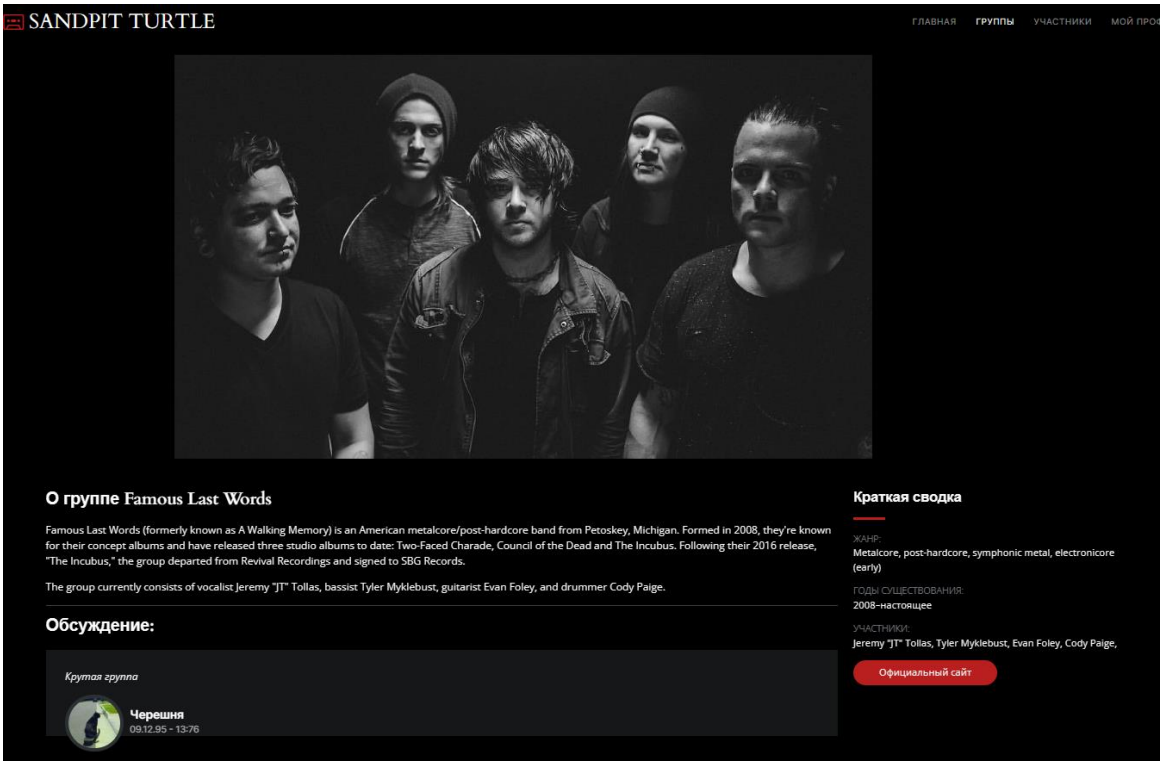

Рис.3.5 Клиентский интерфейс: страница обсуждения

Заключение

В результате выполнения курсовой работы была изучена специальная литература, описаны теоретические аспекты и раскрыты ключевые понятия исследования. В результате проделанной работы было создано клиент-серверное приложение. Форум имеет направление в развлекательной индустрии, однако форумы также используются как источник распространения, то есть дальнейшего маркетинга, обсуждаемых тем, в данном случае – музыкальных групп.

Для создания форума была построена простая трехступенчатая архитектура. Была реализована база данных и сервис для взаимодействия с БД.

Данное приложение соответствует требованиям задания, то есть оно поддерживает работу с базами данных, клиент имеет возможность управлять некоторыми записями базы данных.

Список литературы

- Леве Джувел «Создание служб Windows Communication Foundation» - Питер – 2008г.
- Осипов Д.Л. «InterBase и Delphi. Клиент-серверные базы данных» - ДМК-Пресс – 2015г.
- А. Швец «Погружение в паттерны проектирования» - Самиздат – 2018г.
- С. А. Орлов «Программная инженерия. Технологии разработки программного обеспечения». 5-е издание обновленное и дополненное – СПб.: Питер – 2016г.
- Сибраро П., Клайс К., Коссолино Ф., Грабнер Й . «WCF 4 Windows Communication Foundation и .NET 4» - Диалектика – 2011г.

Программное приложение

Ссылка на GitHub репозиторий: https://github.com/Mdoodl/kurs