

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»  
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем

Расчётно-графическое задание  
По дисциплине: Технологии Web-программирования  
тема: «Разработка веб приложения»

Выполнила: ст. группы ПВ-192

Куценко Мария

Проверил: Картамышев С.В.

Белгород 2022 г.

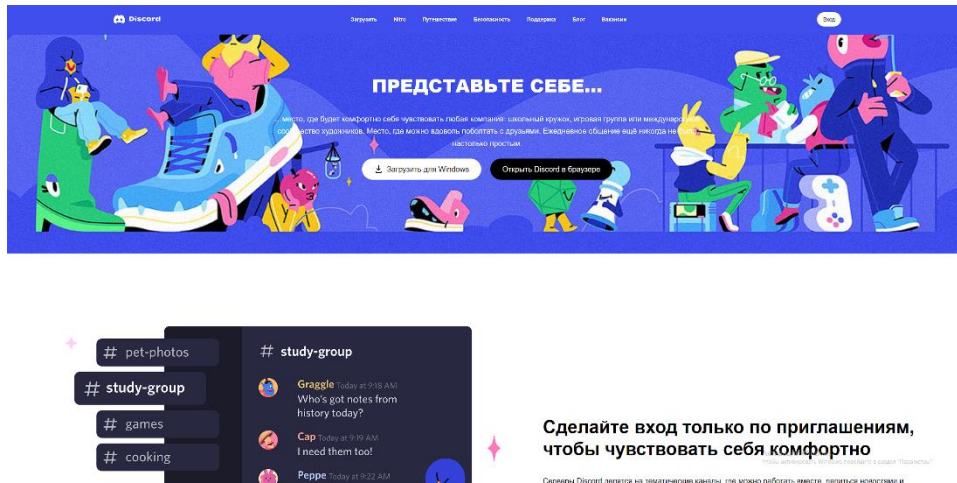
## Оглавление

HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS. ....	3
Клиентское программирование.....	6
Серверное программирование. ....	7
Разработка и проектирование базы данных. ....	8
REST API.....	9
Работа с HTTP запросами.....	10
Заключение.....	10

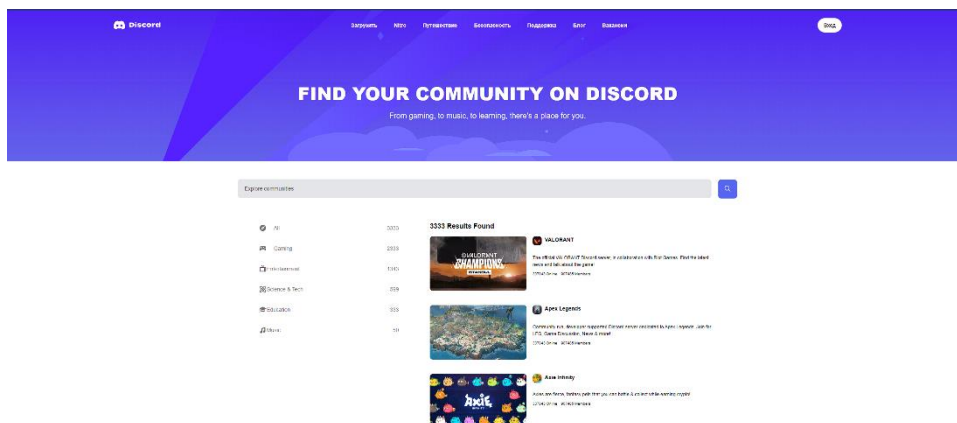
# HTML. Разработка макетов и верстка шаблонов web-приложения с помощью языков HTML и CSS.

В ходе выполнения работы были разработаны следующие страницы, копирующие дизайн сайта discord.com:

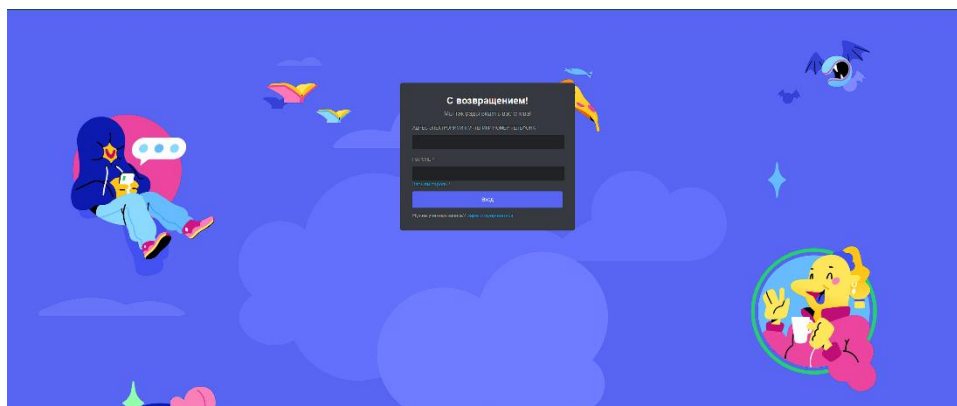
## Главная страница



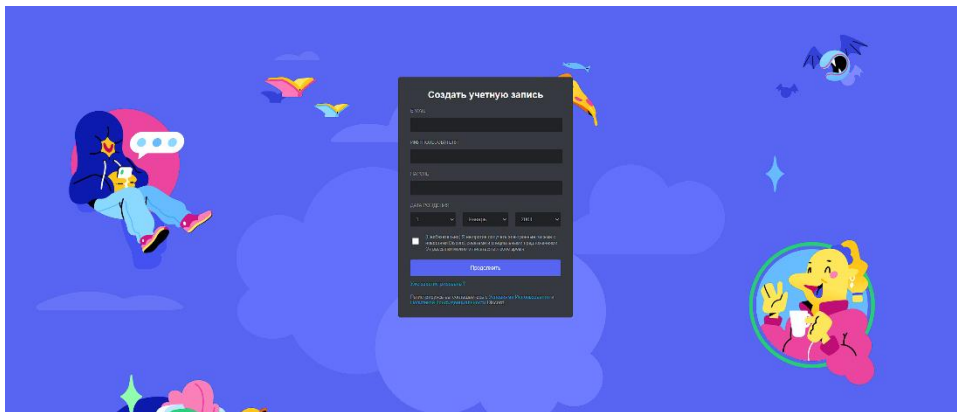
## Страница “Путешествия”



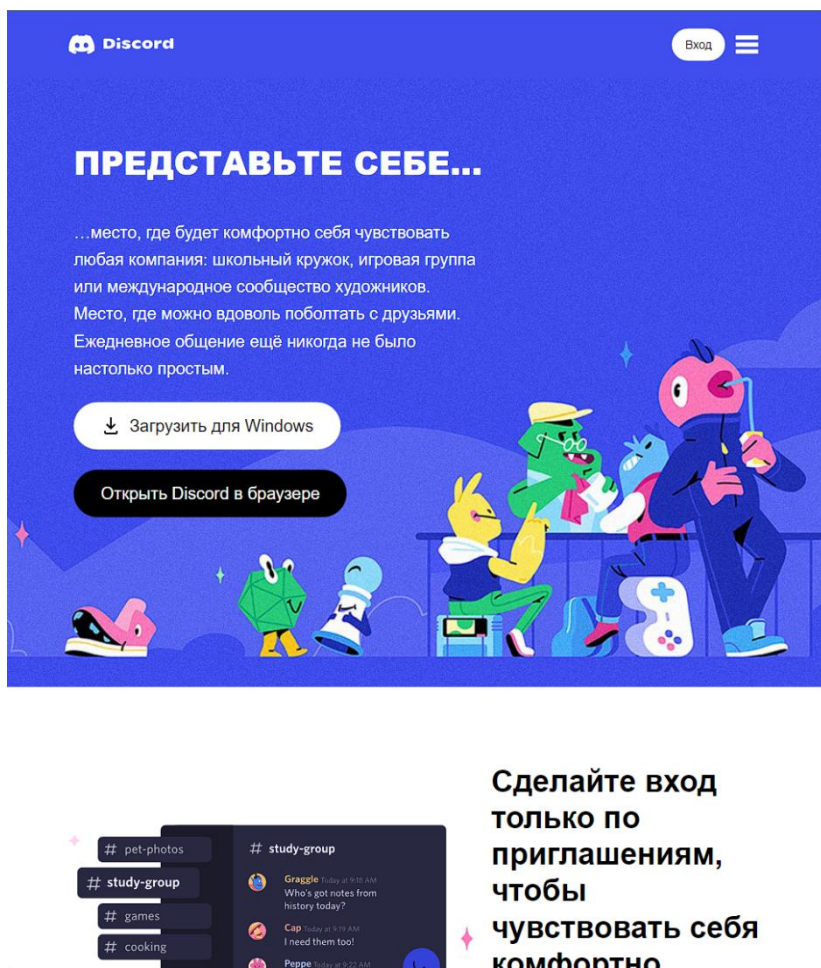
## Страница входа



## Страница регистрации



Были переменены технологии адаптивной верстки:



Пример исходного кода для страницы входа:

login.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
  <link rel="stylesheet" href="../../styles/login.css">
  <script type="text/javascript" src="../../scripts/login.js"></script>

</head>
<body>

<div class="login-block">
  <div>
    <h1 class="hello-text">С возвращением!</h1>
    <div class="glad-to-see-text" >
      Мы так рады видеть вас снова!
    </div>
  </div>

  <form name="login" action="login.html" >
    <div>
      <div>
        <label for="email">
          АДРЕС ЭЛЕКТРОННОЙ ПОЧТЫ ИЛИ НОМЕР ТЕЛЕФОНА
          <span>*</span>
        </label>
        <input type="email" name="email" id="email" >
      </div>

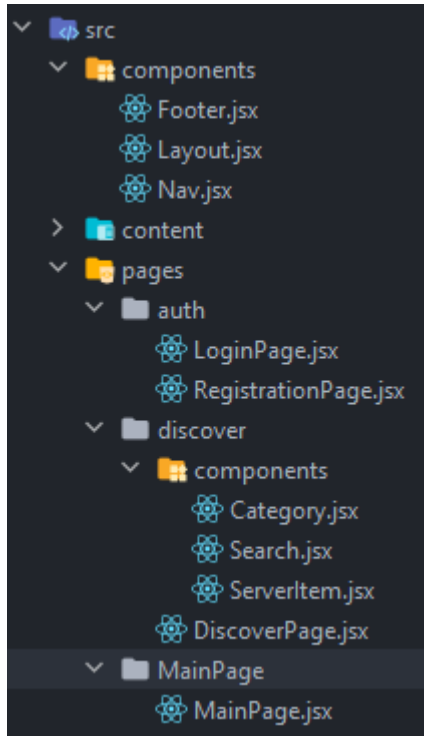
      <div>
        <label for="password">
          ПАРОЛЬ
          <span>*</span>
        </label>
        <input type="password" name="password" id="password" >
        <a href="#" class="inks"> Забыли пароль? </a>
      </div>
    </div>

    <div>
      <button onclick="validateForm()" type="button" class="button" >Вход</button>
      <div>
        <div class="need-registration"> Нужна учетная запись?
          <a href="registration.html" class="inks"> Зарегистрироваться </a>
        </div>
      </div>
    </div>
  </form>
</div>
</body>
</html>

```

## Клиентское программирование.

В ходе выполнения работы были реализовано веб приложение с помощью библиотеки React, в него были перенесены страницы и разбиты на компоненты:






















Пример исходного кода для компонента категорий на странице «путешествия»:

### Category.jsx

```
function Category({props}) {  
  
  const dispatch = useDispatch()  
  const {currentCategory} = useSelector(selector => state.servers)  
  
  return <div className="discover-coll">  
    {  
      props.map((item, key) => {  
        return <button onClick={() => currentCategory !== item.id?  
          dispatch(getServerByCategory({arg: {typeId: item.id}}): null} key={key} className="discover-category-b  
          <svg aria-hidden="false" width="20" height="20" viewBox="0 0 24 24">  
            <g fill="none" fillRule="evenodd">  
              <path fill="currentColor"  
                d="M5.79335761,5 L18.2066424,5 C19.7805584,5 21.0868816,6.21634264 21.1990185,7.7862  
              <rect width="24" height="24"/>  
            </g>  
          </svg>  
          <div className="discover-category-name">{item.name}</div>  
          <div className="discover-category-count">{item.serverCount}</div>  
        </button>  
      })  
    }  
  </div>  
}  
  
export default Category
```

## Серверное программирование.

Произвели контейнеризацию фронтенд части, бэкенд части и базы данных.

	NAME	IMAGE	STATUS	PORT(S)	STARTED	ACTIONS
	discord 3 containers	-	Running (3/3)	-		  
	frontend-1 15043bc247c9 	discord-frontend:latest	Running	3000	13 minutes ago	  
	server-1 a5bec89c0e51 	discord-server:latest	Running	4000	13 minutes ago	  
	db-1 24a815326250 	mysql:latest	Running	3306	13 minutes ago	  

### docker-compose.yaml

```
services:
  server:
    build:
      context: ./server/
    command: npm run dev
    restart: unless-stopped
    environment:
      - DATABASE_DB=discord
      - DATABASE_USER=root
      - DATABASE_PASSWORD=123)Masha
      - DATABASE_HOST=mysqldb
      - DATABASE_PORT=3306
    stdin_open: true
    ports:
      - '4000:4000'
    volumes:
      - ./server:/discord
      - /discord/node_modules
    depends_on:
      - db
    links:
      - db
  db:
    image: mysql
    restart: unless-stopped
    volumes:
      - db-data:/var/lib/mysql
    environment:
      - MYSQL_USER=root
      - MYSQL_PASSWORD=123)Masha
      - MYSQL_DATABASE=discord
      - MYSQL_ROOT_PASSWORD=123)Masha
    ports:
      - "3306:3306"
  frontend:
    build:
      context: client
    ports:
      - 3000:3000
    volumes:
      - ./client:/discord
      - /discord/node_modules
    depends_on:
      - server

volumes:
  back-notused:
  db-data:
```

server/Dockerfile

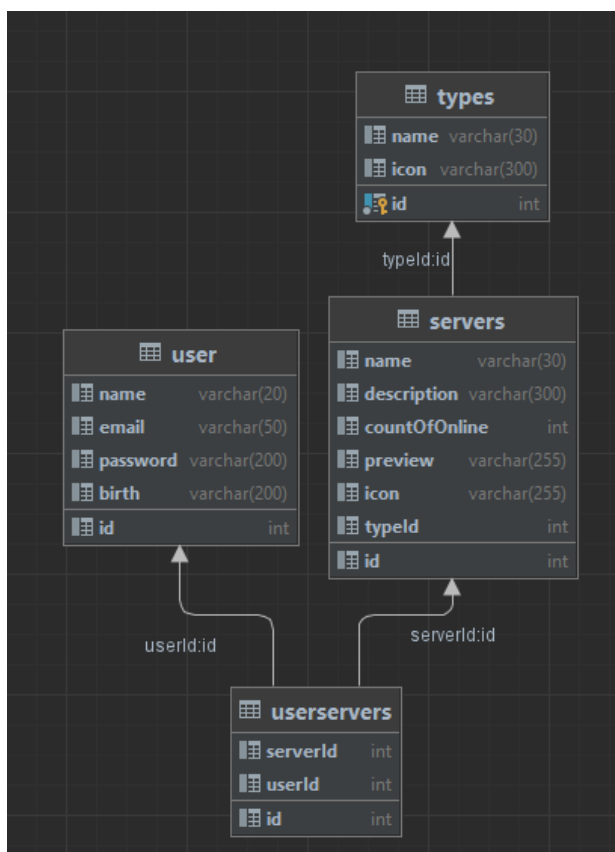
```
FROM node:18
WORKDIR /discord
COPY package.json .
RUN npm install
COPY . .
EXPOSE 4000
CMD ["npm", "run", "server"]
```

client/Dockerfile

```
FROM node:18
WORKDIR /discord
COPY package.json .
RUN npm install
COPY . .
EXPOSE 3000
CMD ["npm", "start"]
```

Разработка и проектирование базы данных.  
В ходе работы была выбрана СУБД MySQL.

Спроектирована следующая структура базы данных:





И определены модели с помощью ORM Sequelize.

Пример исходного кода:

User.js

```
const User = sequelize.define( modelName: 'user', attributes: {  
  
  id: {  
    type: DataTypes.INTEGER,  
    allowNull: false,  
    autoIncrement: true,  
    primaryKey: true,  
    unique: true  
  },  
  name: {  
    type: DataTypes.STRING( length: 20),  
    allowNull: true,  
    unique: false  
  },  
  email: {  
    type: DataTypes.STRING( length: 50),  
    allowNull: false,  
    unique: true  
  },  
  password: {  
    type: DataTypes.STRING( length: 200),  
    allowNull: false  
  },  
  birth: {  
    type: DataTypes.STRING( length: 200),  
    // type: DataTypes.DATEONLY,  
  }  
}, options: {  
  timestamps: false,  
  tableName: 'user'  
})  
  
module.exports = {User}
```

## REST API.

Была изучена структура формата представления данных JSON, типы запросов к API: HEAD, GET, POST, PUT, DELETE, спроектировано и реализовано собственное REST API.

Схема API Swagger

Авторизация		^
POST	/registration	▼
POST	/login	▼
GET	/me	▼
Сервера		^
GET	/discover	▼
GET	/discover/search	▼
GET	/discover/filter	▼
PUT	/discover	▼

## Работа с HTTP запросами.

В качестве библиотеки для работы с аях запросами был выбран axios.

Пример исходного кода:

Запрос на добавление пользователя на discord-сервер

ServerSlice.js

```
export const addUserToServer = createAsyncThunk(
  'servers/addUserToServer',
  payloadCreator.async ({serverId}, serverAPI) => {
    try {
      const {data} = await axios.put('url: /discover', {data: {serverId}})

      serverAPI.dispatch(setCurrentServers(serverId))

      return data
    } catch (e) {
      console.log(e)
    }
  }
)
```

```
reducers: {
  setCurrentServers:(state: Draft<State>, action: PayloadAction<any>)=>{
    state.currentServers.includes(action.payload)?

    state.currentServers.splice(state.currentServers.indexOf(action.payload), deleteCount: 1):
    state.currentServers.push((action.payload))
  }
},
```

## Заключение

Благодаря описанным действиям получилось реализовать веб приложения с возможностью развёртывания на удаленном хостинге, на котором авторизованные пользователи могут добавляться или уходить с каналов discord.