Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

“Онлайн сервис продажи билетов на спортивные матчи”

Выполнили студенты гр. 853504

Пресный В.И.

Грищенко М.М.

Кузьма В.В

Проверил

Чащин С.В.

Минск, 2021

**1.ВВЕДЕНИЕ**

GoTicket представляет собой приложение для покупки билетов на предстоящие матчи. Основным назначением разработки является создание сервиса продажи билетов на различные события.

**1.1.ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ**

Процессор: Pentium 4 @ 3 ГГц/AMD 64 3200+;

Оперативная память: 512 МБ (1 ГБ для Vista);

Жесткий диск: 8 ГБ свободного места;

**1.2.ТРЕБОВАНИЯ К ПРОГРАММНОЙ СОВМЕСТИМОСТИ**

Требования к операционной системе: сервер - Windows, клиент - любая операционная система.

**2.ПРЕДМЕТНАЯ ОБЛАСТЬ**

Приложение для покупки билетов на различные спортивные мероприятия в странах СНГ.

**3.СРЕДСТВА РЕАЛИЗАЦИИ**

Python PyQt5 +SQLite3.

**4.ОПИСАНИЕ ФУНКЦИОНАЛА**

4.1 Роли пользователей — пользователь может иметь несколько ролей

4.1.1. Обычный пользователь(User)

4.1.1.1 Незарегистрированный пользователь может просматривать матчи и информацию о самих матчах.

4.1.1.2. Зарегистрированный пользователь может покупать билеты на интересующие матчи. Выполнять поиск определенных матчей или выводить список матчей на интересующий спорт.

4.1.2. Модератор (Moderation) обладает всем функционалом обычного пользователя, но имеет дополнительные возможности, а именно:

4.1.2.1. Просмотр статистики пользователей (основная информация, дата регистрации, купленный билеты)

4.1.2.2. Создание матчей

4.1.3. Администратор сайта (Admin) обладает всем функционалом модератора с дополнительными привилегиями:

4.1.3.1. Управление ролями (выдача роли модератора)

4.2. Авторизация, регистрация пользователей

4.3. Журналирование действий пользователей

**5.СВЯЗИ МЕЖДУ СУЩНОСТЯМИ**

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность | Кардинальность | Связующая сущность | Номер связи |
| event | 1:M | ticket | 1 |
| event | 1:1 | event\_info | 2 |
| event | 1:M | address | 3 |
| user | 1:1 | user\_info | 4 |
| user | 1:1 | role | 5 |
| user | 1:M | ticket | 6 |

|  |  |
| --- | --- |
| Номер связи | Описание |
| 1 | На одно событие может быть много билетов. Множество курсов могут относится к одному событию |
| 2 | Одно события может иметь одну информацию о себе. Информация может относится только к одному событию |
| 3 | По одному адресу могут проводится много мероприятий. Множество мероприятий могут проводится по одному адресу |
| 4 | Один пользователь может иметь одну информацию о себе. Одна информация может относится только к одному пользователю |
| 5 | Один пользователь может иметь одну роль. Одна роль может принадлежать одному пользователю |
| 6 | Один пользователь может иметь много билетов. Множество билетов могут принадлежать одному пользователю |

1. **М:М**  -> человек - событие(человек можно сходить на неограниченное количество событий, на событии может быть неограниченное число человек);
2. **Рекурсивная** -> роль - роль (каждый администратор - модератор, каждый модератор - пользователь);
3. **Идентифицирующая** -> событие - место проведения ( без события нет места проведения), событие - билет(без события нет возможности приобрести билет), человек - билет(только человек может приобрести билет),событие - дополнительная информация(событие содержит дополнительную информацию), пользователь - пользовательская информация (пользовательская информация содержит информацию только о пользователе);

**Приложение 1**

Ссылка на ER диаграмму

<https://ibb.co/pQgwb7C>

**Приложение 2**

Создание таблицы “personal\_info

CREATE TABLE IF NOT EXISTS personal\_info(

id integer primary key autoincrement not null,

fio varchar(128) not null,

age integer not null

)

Создание таблицы “user”

CREATE TABLE IF NOT EXISTS USER(

id integer primary key autoincrement not null,

password VARCHAR(200) not null,

login VARCHAR(200) not null,

registration\_date date not null,

role\_id integer,

personal\_info\_id integer,

foreign key (role\_id) references role(id),

foreign key (personal\_info\_id) references personal\_info(id) ON DELETE CASCADE

);

Создание таблицы “role”

CREATE TABLE "role" (

"id" integer NOT NULL,

"name" TEXT,

"boss" TEXT,

PRIMARY KEY("id" AUTOINCREMENT),

FOREIGN KEY("boss") REFERENCES role

);

Создание таблицы “country”

CREATE TABLE IF NOT EXISTS country(

id integer primary key autoincrement not null,

name varchar(128)

)

Создание таблицы “city”

CREATE TABLE IF NOT EXISTS city(

id integer primary key autoincrement not null,

name varchar(128)

)

Создание таблицы “street”

CREATE TABLE IF NOT EXISTS street(

id integer primary key autoincrement not null,

name varchar(128)

)

Создание таблицы “address”

CREATE TABLE IF NOT EXISTS address(

id integer primary key autoincrement not null,

country\_id tinyint,

city\_id integer,

street\_id integer,

foreign key (country\_id) references country(id),

foreign key (city\_id) references city(id),

foreign key (street\_id) references street(id)

)

Создание таблицы “ticket”

CREATE TABLE IF NOT EXISTS ticket(

id integer primary key autoincrement not null,

date date not null,

user\_id integer,

event\_id integer,

foreign key (user\_id) references user(id),

foreign key (event\_id) references event(id)

);

Создание таблицы “event\_info”

CREATE TABLE IF NOT EXISTS event\_info(

id integer primary key autoincrement not null,

date date not null,

duration smallint not null,

description varchar(300)

);

Создание таблицы “event”

CREATE TABLE IF NOT EXISTS event(

id integer primary key autoincrement not null,

name varchar(100) not null,

ticket\_amount smallint,

event\_info\_id integer,

address\_id integer,

foreign key (event\_info\_id) references event\_info(id) ON DELETE CASCADE,

foreign key (address\_id) references address(id)

);

**Приложение 3**

Получение всех событий

select \* from event;

Получение записей из таблицы “event\_info” с описанием, начинающимся по маске.

select \* from event\_info where description LIKE ?;

Получение роли пользователя

SELECT role.name FROM user JOIN role ON user.role\_id = role.id WHERE user.login = ?;

Получение всех пользователей с определенной ролью

select \* from user where role\_id = ?;

Получение ролей с именем.

select \* from role where name = ?;

Получение ролей с определенным id.

select \* from role where id = ?;

Получение всех билетов определенного пользователя.

select \* from ticket where user\_id = ?;

Получение всех билетов на определенное событие

select \* from ticket where event\_id = ?;

Получение определенной страны.

select \* from country where id = ?;

Получение записей из таблицы “event\_info” с определенным описанием.

select \* from event\_info where description = ?;

Получение определенного города.

select \* from city where id = ?;

Получение определенной улицы

select \* from street where id = ?;

Получение определенной страны

select \* from country where name = ?;

Получение определенной города

select \* from city where name = ?;

Получение определенной улицы

select \* from street where name = ?;

Получение определенного адреса

select \* from address where id = ?;

Получение определенного адреса по стране

select \* from address where country\_id = ?;

Получение определенного адреса по городу

select \* from address where city\_id = ?;

Получение определенного адреса по улице

select \* from address where street\_id = ?;

Получение всех событий по имени

select \* from event where name = ?;

Получение определенной информации об ивенте

select \* from event\_info where id = ?;

;

Получение всех имен пользователей

select login from user;

Получение всех персональных данных о пользователях

select personal\_info\_id from user;

Получение пароля пользователя

select password from user where login = ?;

Получение всех пользователей с определенным именем

select \* from user where login = ?;

Получение всех ивентов с именем по маске

SELECT \* FROM event WHERE mylower(name) LIKE ?

Получение всех ивентов с определенной информацией

SELECT \* FROM event WHERE event\_info\_id = ?

Получение id последней созданной персональной информации

SELECT id FROM personal\_info ORDER BY id DESC LIMIT 1'''

Получение списка ролей для рекурсивной связи

WITH RECURSIVE

under\_role(name) AS (

VALUES(?) --VALUES(?)

UNION ALL

SELECT role.name FROM role JOIN under\_role ON role.boss = under\_role.name

)

SELECT name FROM under\_role;

**Приложение 4**

Добавление новой сущности в таблицу “country”

INSERT INTO country(name) VALUES (?);

Добавление новой сущности в таблицу “street”

INSERT INTO street (name) VALUES (?);

Добавление новой сущности в таблицу “city”

INSERT INTO city (name) VALUES (?);

Добавление новой сущности в таблицу “address”

INSERT INTO address (country\_id, city\_id, street\_id) VALUES (?,?,?);

Добавление новой сущности в таблицу “ticket”

INSERT INTO ticket (price, date, user\_id, event\_id) VALUES (?,?,?,?,?);

Добавление новой сущности в таблицу “event”

INSERT INTO event (name, ticket\_amount, event\_info\_id, address\_id) VALUES (?,?,?,?);

Добавление новой сущности в таблицу “personal\_info”

INSERT INTO personal\_info (fio, age) VALUES (?,?);

Добавление новой сущности в таблицу “role”

INSERT INTO role (name, role\_id) VALUES (?,?);

Добавление новой сущности в таблицу “event\_info”

INSERT INTO event\_info (description, duration, date) VALUES (?,?, ?);

Добавление новой сущности в таблицу “user”

INSERT INTO user (password, login, registration\_date, balance, role\_id, personal\_info\_id) VALUES (?, ?,?, ?,?,?);

**Приложение 5**

Триггер журналирует создание пользователя

CREATE TRIGGER IF NOT EXISTS log\_insert\_user

AFTER INSERT

ON USER

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (NEW.id, -1, -1, DATE(), 'INSERT');

END;

Триггер журналирует обновление пользователя

CREATE TRIGGER IF NOT EXISTS log\_update\_user

AFTER UPDATE

ON USER

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (OLD.id, -1, -1, DATE(), 'UPDATE');

END;

Триггер журналирует удаление пользователя

CREATE TRIGGER IF NOT EXISTS log\_delete\_user

AFTER DELETE

ON USER

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (OLD.id, -1, -1, DATE(), 'DELETE');

END;

Триггер журналирует создание билета

CREATE TRIGGER IF NOT EXISTS log\_insert\_ticket

AFTER INSERT

ON TICKET

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (-1, NEW.id, -1, DATE(), 'INSERT');

END;

Триггер журналирует обновление билета

CREATE TRIGGER IF NOT EXISTS log\_update\_ticket

AFTER UPDATE

ON TICKET

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (-1, OLD.id, -1, DATE(), 'UPDATE');

END;

Триггер журналирует удаление билета

CREATE TRIGGER IF NOT EXISTS log\_delete\_ticket

AFTER DELETE

ON TICKET

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (-1, OLD.id, -1, DATE(), 'DELETE');

END;

Триггер журналирует создание события

CREATE TRIGGER IF NOT EXISTS log\_insert\_event

AFTER INSERT

ON EVENT

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (-1, -1, NEW.id, DATE(), 'INSERT');

END;

Триггер журналирует обновление события

CREATE TRIGGER IF NOT EXISTS log\_update\_event

AFTER UPDATE

ON EVENT

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (-1, -1, OLD.id, DATE(), 'UPDATE');

END;

Триггер журналирует удаление события

CREATE TRIGGER IF NOT EXISTS log\_delete\_event

AFTER DELETE

ON EVENT

BEGIN

INSERT INTO log\_table(user\_id, ticket\_id, event\_id, operation\_time, operation\_action)

VALUES (-1, -1, OLD.id, DATE(), 'DELETE');

END;

**Приложение 6**

Проволочные интерфейсы:

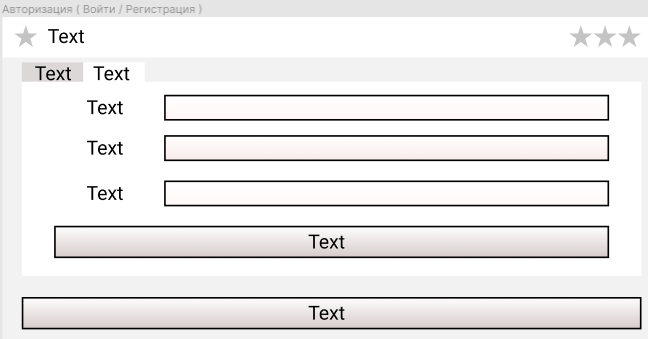


Рис. 1. Страница регистрации

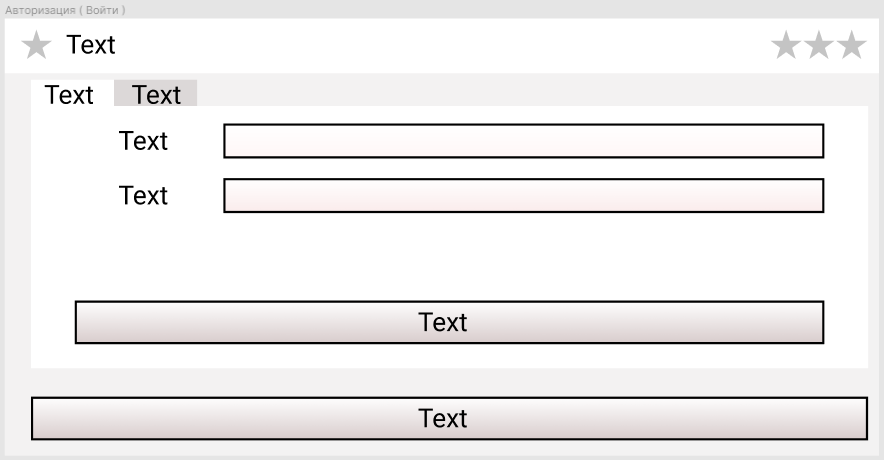


Рис. 2. Страница авторизации

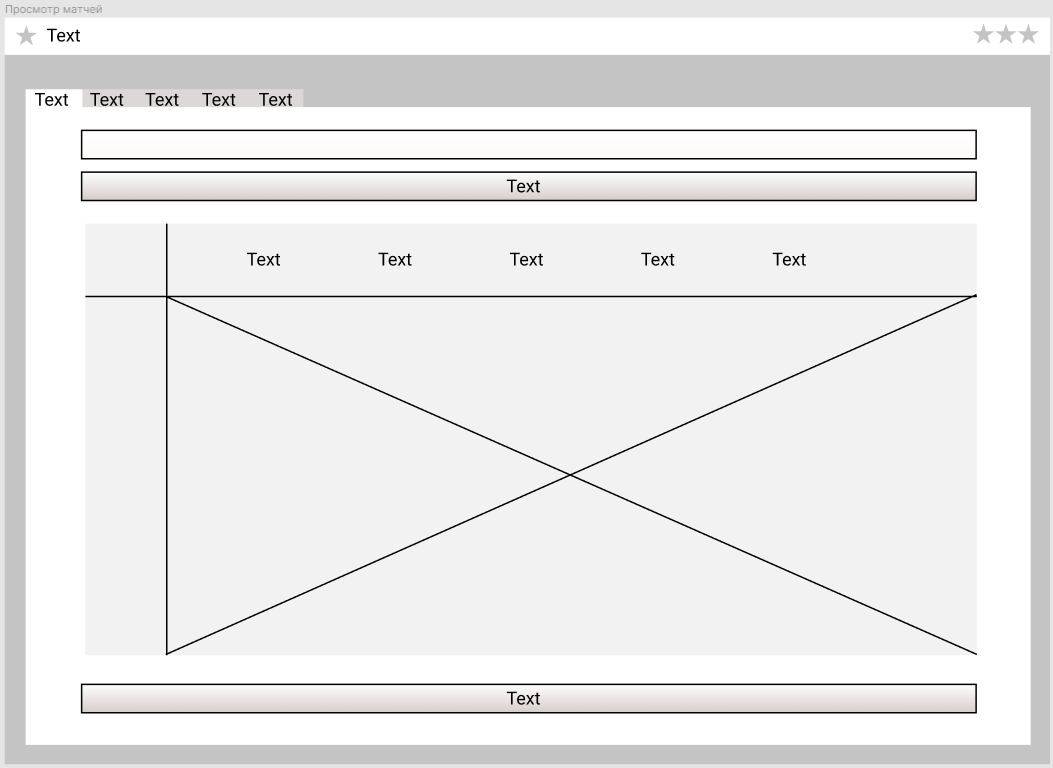


Рис. 3. Страница с матчами

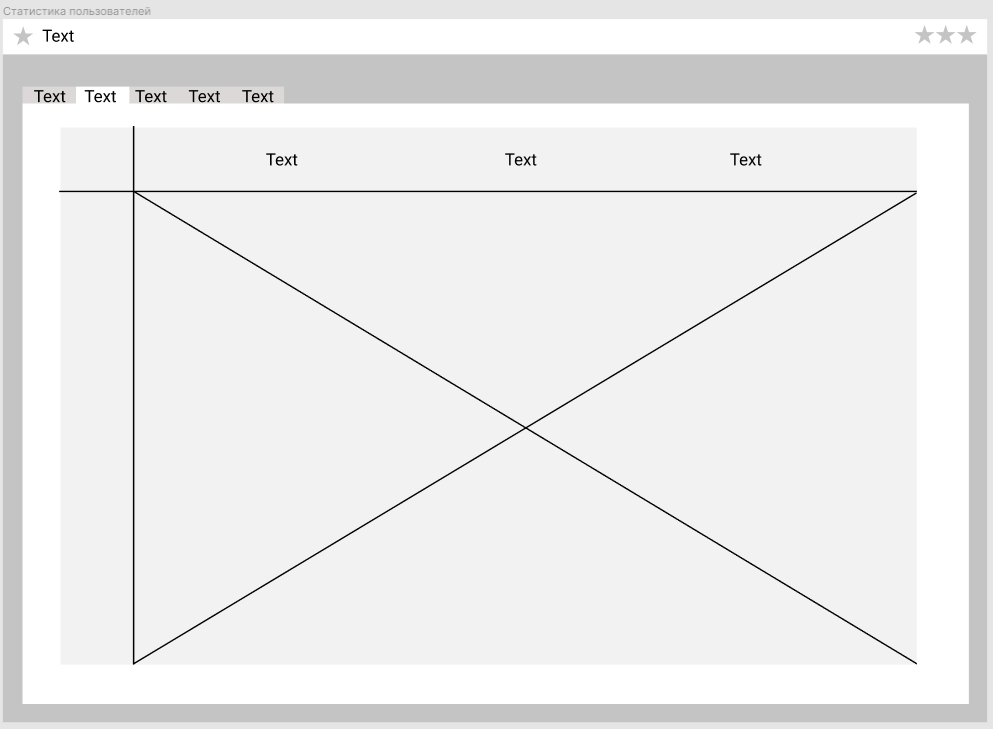


Рис. 4. Страница статистики пользователей

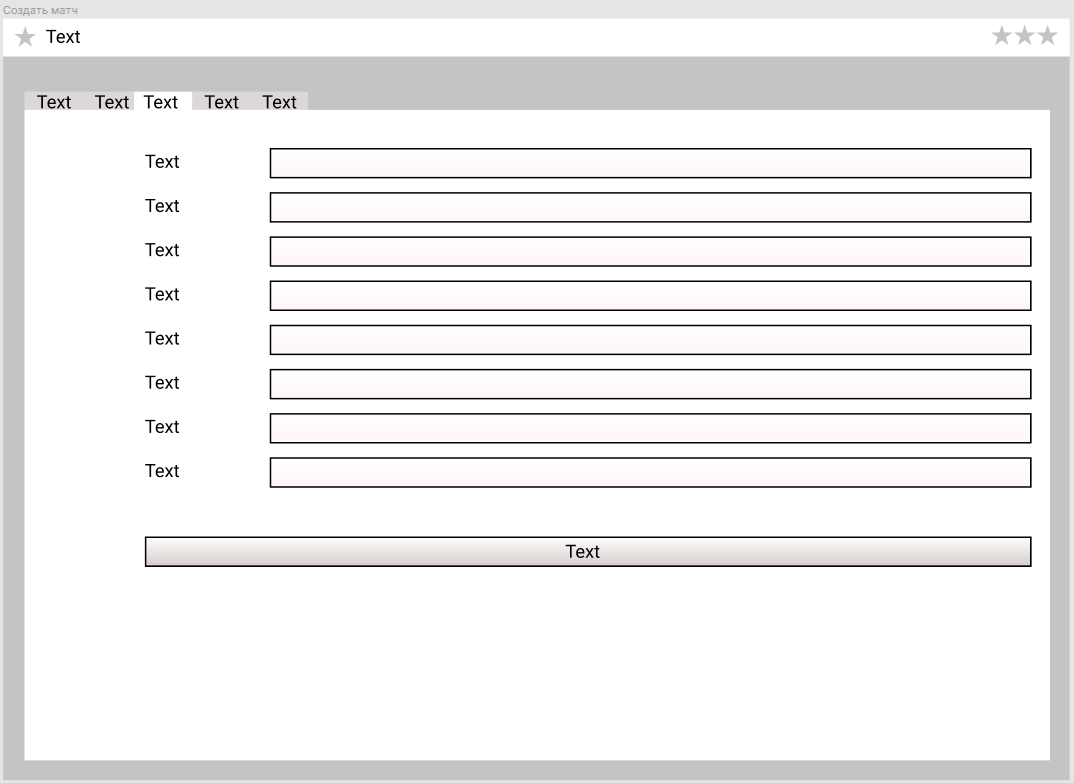


Рис. 5. Страница создания матча

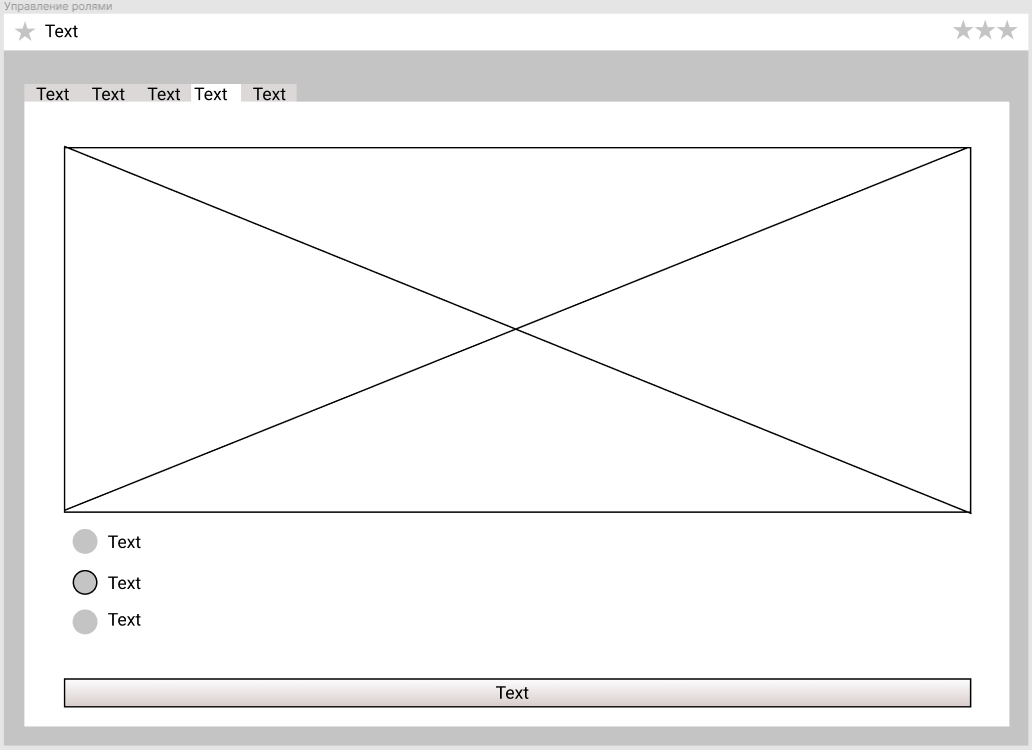


Рис. 6. Страница управления пользователями

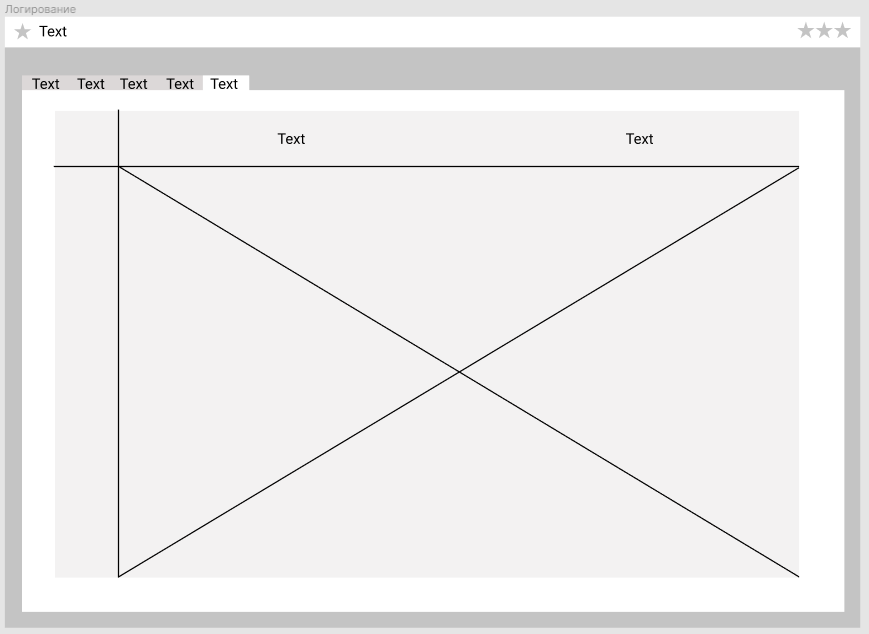


Рис. 7. Страница логирования

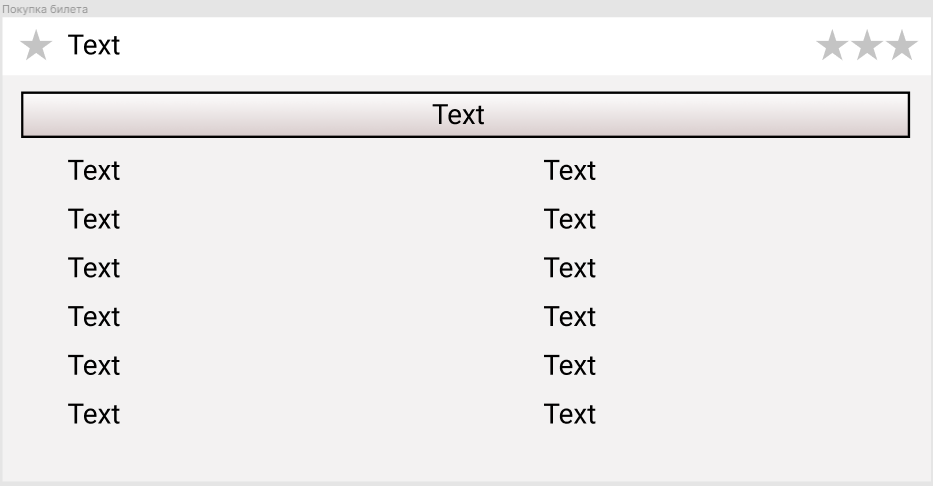


Рис. 8. Страница детальной информации о матче

Индексы используются для увеличения производительности БД, но есть случаи, когда нам стоит избегать их использования:

Не стоит использовать индексы для небольших таблиц.

Не стоит использовать индексы для таблиц, в которых, как предполагается, будут часто добавляться новые данные, либо эти данные будут изменяться.

Не стоит использовать индекс для колонок, с которыми будут производиться частые манипуляции.

Не стоит использовать индексы для колонок, которые имеют много значений NULL.

В таблицах с небольшим количеством полей может не использоваться индекс, если из нее всегда извлекается большой процент записей. Создание индекса не предотвратит полное сканирование таблиц. Основные операции ввода-вывода в операционках считываются в блоках или страницах.Важно понимать, что если таблица имеет достаточно небольшое количество записей, чтобы занимать одну физическую страницу, зачем читать индекс и указывать на таблицу? Чтение индекса и таблицы-это чтение двух страниц, когда чтение таблицы представляет собой ввод-вывод только на одной странице. Индексы обычно должны создаваться на небольшом проценте полей в таблице. Большие составные индексы могут быть относительно большими по сравнению с таблицей. Важен относительный размер между индексом и таблицей. Чем больше отношение индекса к физическому размеру таблицы, тем менее полезным будет индекс с точки зрения уменьшения физического пространства для чтения. Возможно, будет быстрее прочитать всю таблицу, а не большой составной индекс поля

Исходя из вышесказанного, использование индексов в нашей бд необязательно

**Приложение 7**

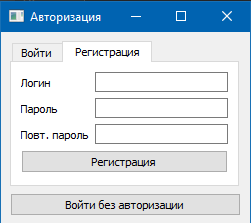


Рис 9. Окно регистрации

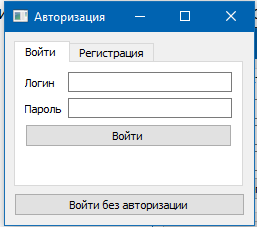


Рис 10. Окно авторизации

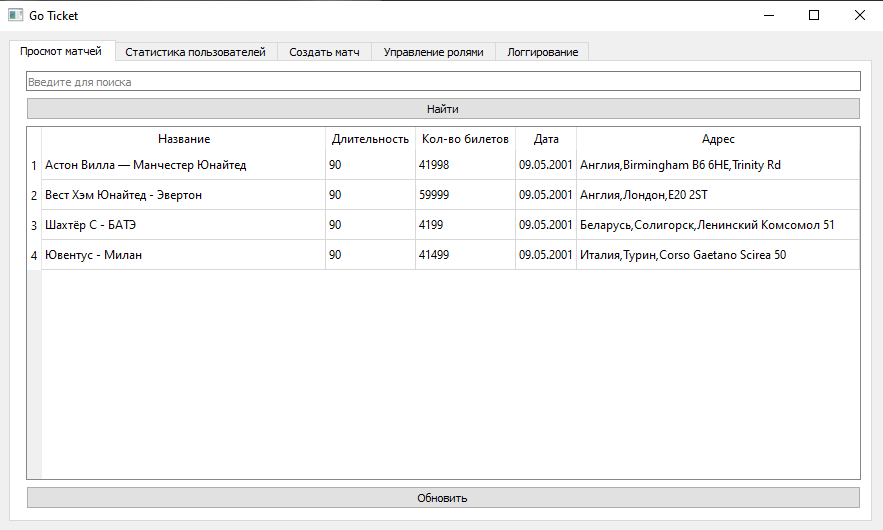


Рис 11. Просмотр матчей

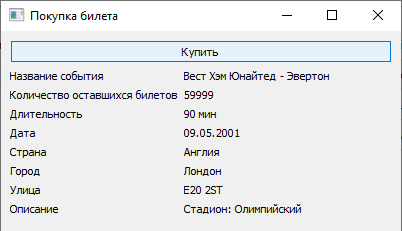


Рис 12. Детальный просмотр матча и покупка билета.

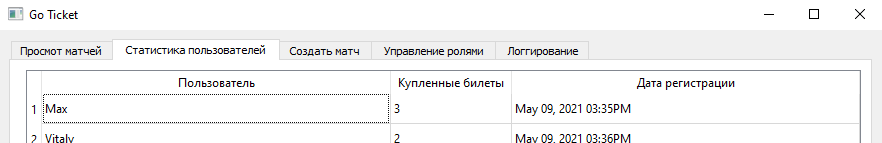


Рис 13. Статистика пользователей

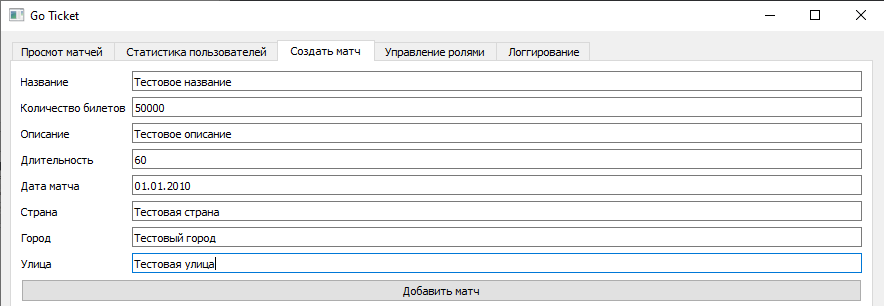


Рис 14. Создание матча

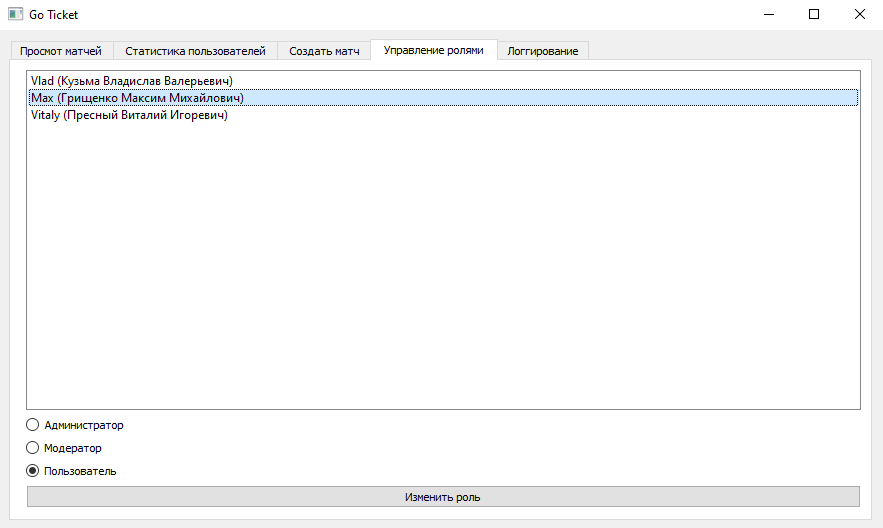


Рис 15. Управление ролями

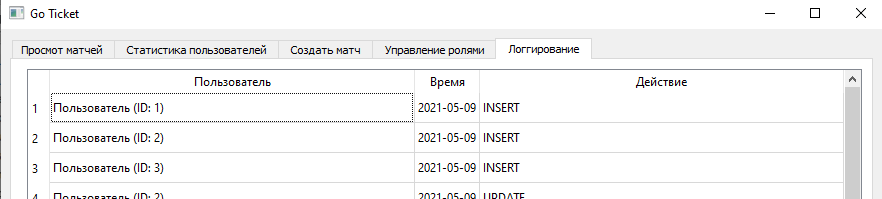


Рис 16. Логгирование