

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**

---



**Институт  
интеллектуальных кибернетических систем  
Кафедра №22 «Кибернетика»**

Направление подготовки 09.04.04 Программная инженерия

## **ОТЧЕТ О КУРСОВОЙ РАБОТЕ**

По дисциплине “Современные средства программирования”

**Преподаватель:** Садчиков Сергей Михайлович

**Студент:** Космынин Максим Андреевич

**Группа:** Б20-524

## Оглавление

1.	Формулировка задания .....	3
2.	Описание предметной области.....	4
2.1	Условности и ограничения в прототипе .....	4
2.1	Проектирование БД.....	5
2.2	еЕРС диаграммы.....	6
2.3	Функциональная модель .....	10
2.4	Физическая модель БД.....	11
3.	Реализация.....	11
3.1	Создание и заполнение базы данных.....	11
3.2	Описание структуры проекта и используемых технологий .....	15
3.3	Соответствие требованиям .....	18
4.	Демонстрация .....	19
4.1	Демонстрация функционала незарегистрированного пользователя. ....	19
4.2	Демонстрация функционала клиента .....	22
4.3	Демонстрация функционала сотрудника.....	27
4.4	Использование методов POST и GET .....	30
4.5	Реализация триггера.....	32
4.6	AJAX запросы.....	33
4.7	Реализация одновременной работы пользователей.....	33
4.8	Демонстрация изменений на стороне клиента.....	36
4.9	Реализация некоторых требований, которые не были указаны в демонстрации .....	37
4.10	Описание решений по интерфейсу.....	39
5	Сценарий демонстрации .....	42
6	Источники .....	49

# 1. Формулировка задания

Необходимо было реализовать web – приложение банка, позволяющее пользователям открывать счета и получать банковские карточки своих счетов.

Сценарий работы:

1. Неавторизованный пользователь может зарегистрироваться или авторизоваться, посещать страницы для входа, регистрации и стартовую страницу.
2. Клиент может отправлять заявки на открытие банковского счета, т.е. заполнить договор. Также клиент может смотреть информацию об своих счетах и банковских карточках.
3. Сотрудник банка может проверять корректность заполнения договоров, обрабатывать их, создавать новых пользователей.

Web-приложение должно включать:

1. Набор динамических страниц PHP с использованием технологий HTML, хотя бы одна страница должна включать CSS , JavaScript, AJAX;
2. Несколько таблиц в MySQL (в одной или более БД), кроме MySQL можно дополнительно использовать другую СУБД (необходимо предварительно согласовать);
3. Home-страницу с несколькими пунктами меню (можно обычные ссылки) и авторизацией;
4. Два типа пользователей с разными правами (например, сотрудник, который может записывать в БД, и неавторизованный пользователь, который только считывает информацию);
5. Страницу (страницы) с вводом (изменением, добавлением) данных как минимум в 2 связанные таблицы БД одновременно (предполагается, что данные связаны, например, в одну таблицу записываются данные о персоне, а в связанную – список контактных телефонов);
6. Вывод данных по разным запросам (возможно, один запрос будет представлен графиком), как минимум один запрос должен быть по нескольким «связанным» таблицам БД; организация «постраничного» вывода таблиц – по желанию.
7. Асинхронный запрос (AJAX) хотя бы к одной таблице БД.

## **2. Описание предметной области**

При анализе предметной области были выделены несколько типов пользователей с различным правами доступа, которые были описаны в Формулировке. Информация о сотрудниках занесена в систему заранее.

При анализе предметной области были изучены функционалы сайтов “Сбербанка” и “Райффайзен банка”. Также были рассмотрены статьи [1] и [2] (пункт 6) по теме открытия банковского счета онлайн. Из анализа видно, что разные банки по разному подходят к предоставлению услуги по открытию банковского счета в онлайн формате. Но web приложения рассмотренных банков сходятся в том, что для получения возможности пользоваться услугами банков, пользователю необходимо иметь или зарегистрировать аккаунт в приложении банка. Исходя из этого, в прототипе, описанном в данном отчете, пользователь также должен иметь свой аккаунт. Проверкой документов в рассматриваемых банках занимается команда специалистов, в данном прототипе эту функцию выполняют сотрудники, которое также занимаются проверкой, но более “урезанный функционал” (см. далее пункт “Ограничения”). Сама процедура открытия счета реализована по примеру рассматриваемых банков [3], а именно для получения услуги банка по открытию счета, клиент должен выбрать параметры желаемого счета и заполнить договор (необходимые данные клиента для заполнения договора были выявлены из выше указанных источников, а также текста договора [4].

### **2.1 Условности и ограничения в прототипе**

Первая условность вводится на этапе регистрации аккаунта, в разных банках этот процесс устроен по разному, но они сходятся в том, что при создании аккаунта необходимо подтвердить почту или номер телефона, в данном прототипе эта проверка опускается.

На этапе регистрации или заполнения договора в рассматриваемых приложениях есть пункт, запрашивающий от пользователя согласие на обработку персональных данных, эта проверка также опускается.

В свойствах договора для упрощения модели опускается поле “Срок действия”, но информация о дате создания договора сохраняется.

В полях паспорта намеренно нет поля “кем выдан” это сделано для экономии времени при тестировании прототипа.

В данном проекте опущена функция контроля сотрудника за выдачей банковской карты, при одобрении счета, клиент получает сообщение о готовности своей карты и что он может её забрать из отделения банка, но при этом в проекте ни как не прописана такая сущность как “отделение банка”.

## 2.1 Проектирование БД

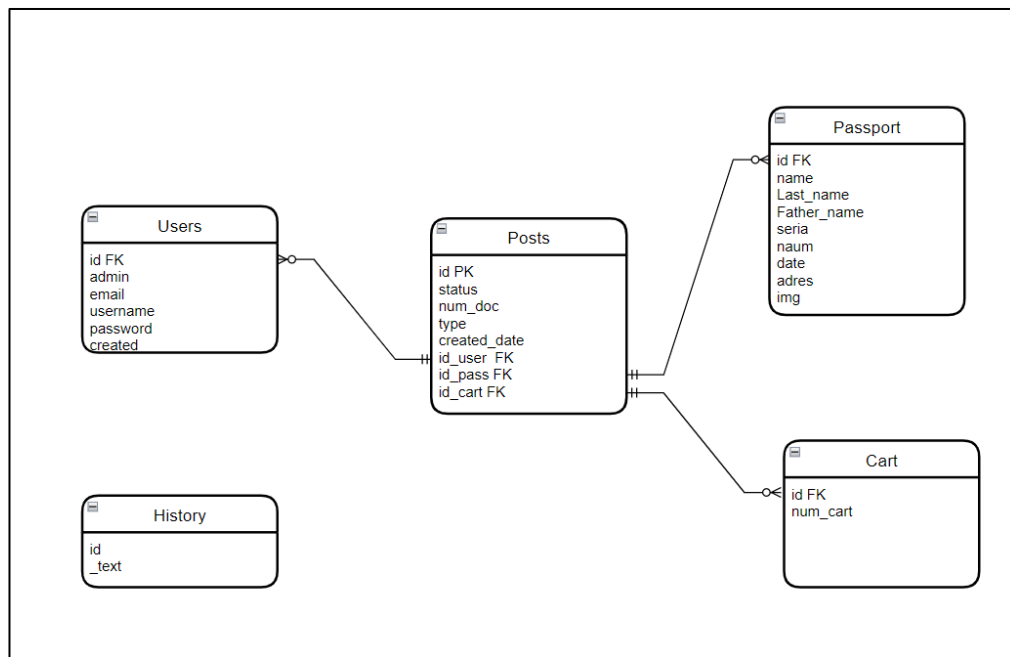


Рис. 1 Логическая модель базы данных - нотация Crow's Foot(Воронья лапка)

Связи:

- Связь Users – Posts. У пользователя может быть от 0 до N договоров, а у договора обязательно есть пользователь.
- Связь Posts –Passport. У договора может быть только один паспорт, а у паспорта есть один договор.
- Связь Posts – Cart. У договора может быть только одна карта, а у карты есть один договор.
- Таблица History не имеет связей, т.к. она служит для реализации триггера в бд (см. далее)

## 2.2 eEPC диаграммы

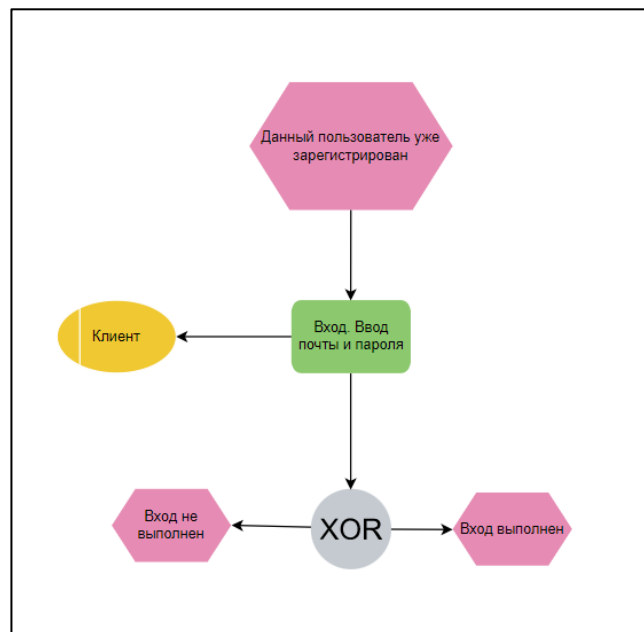


Рис. 1.1 - eEPC диаграмма для авторизации.  
(Extended event driven process chain)

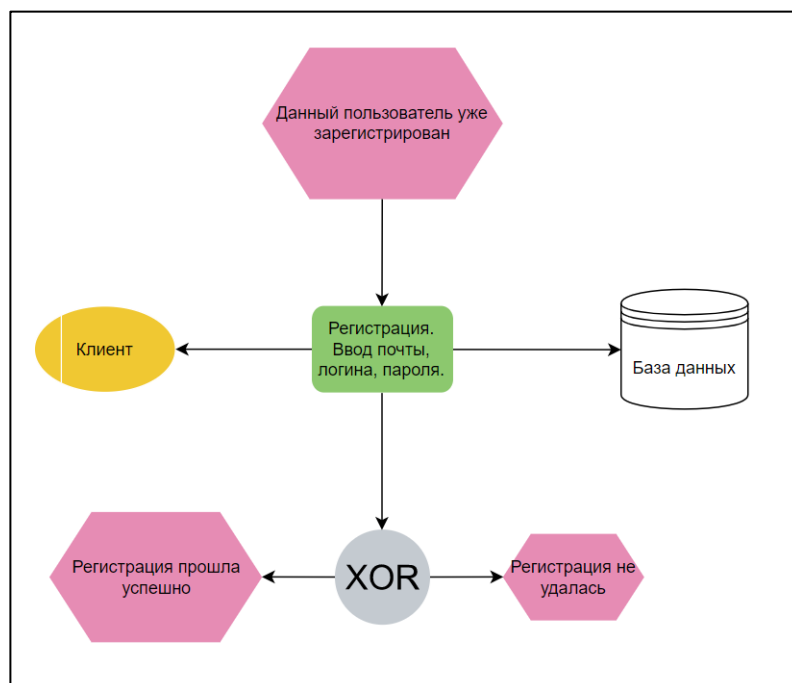


Рис. 1.2 - eEPC диаграмма для регистрации.  
(Extended event driven process chain)

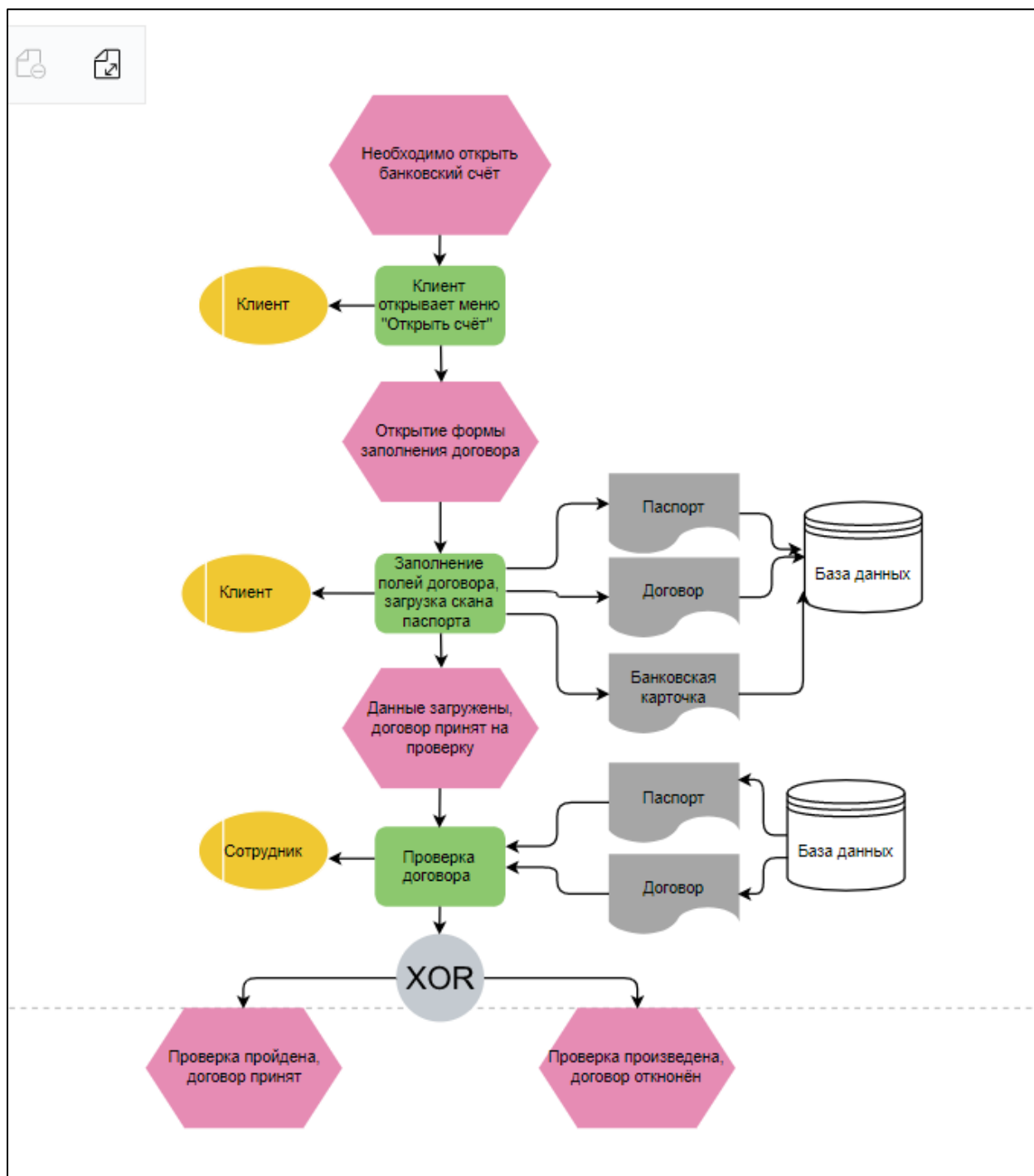


Рис. 1.3 - eEPC диаграмма формирования договора клиентом и проверки его сотрудником.

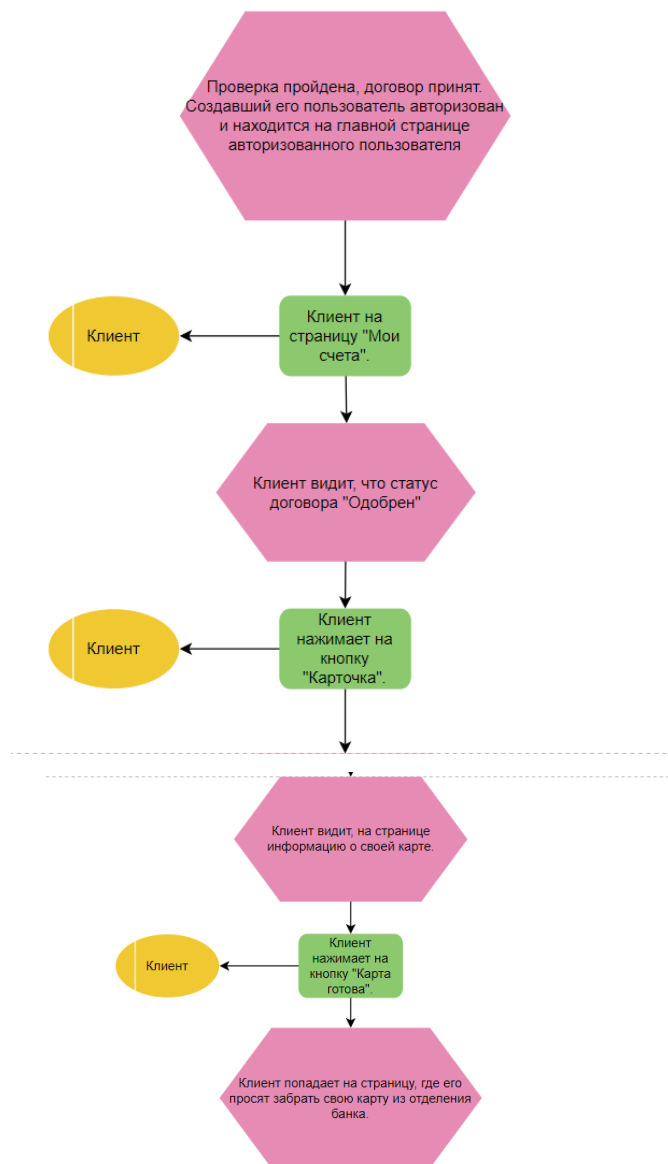


Рис. 1.4 - eEPC диаграмма успешного получения банковской карточки клиентом.



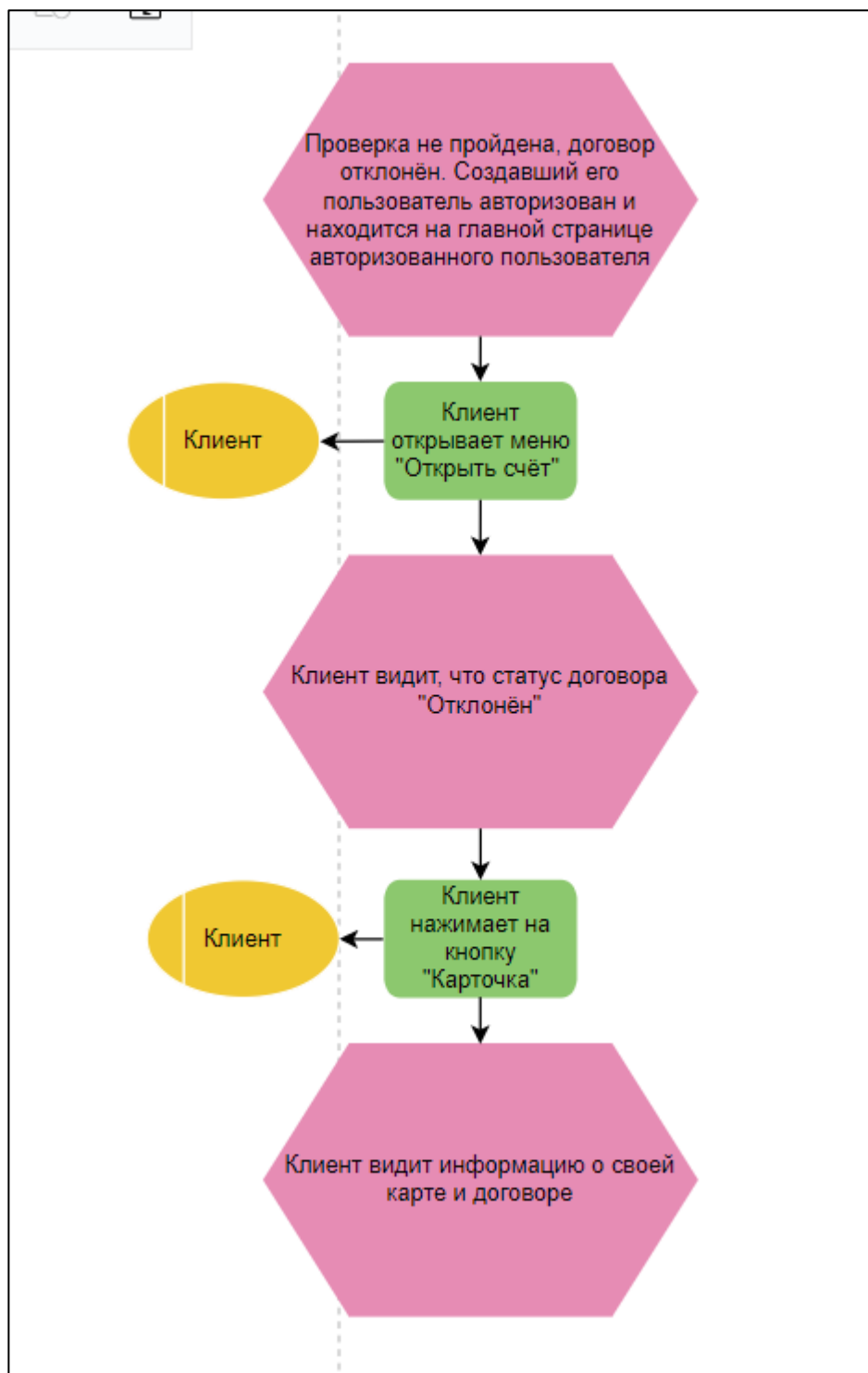


Рис. 1.5 - eEPC диаграмма просмотра информации клиентом о своей банковской карточке при отклонённом договоре.

Пояснение: при создании договора пользователем, автоматически создается его банковская карточка (она делается на основе введенных данных и данных договора). Поэтому пользователь с отклоненным договором может просматривать информацию о карте, но ему не доступен функционал по получению своей карты. Это сделано из предположения, что такой подход побудит пользователя пересоздать договор, так как он

видит, что его карта “почти готова”.

## 2.3 Функциональная модель

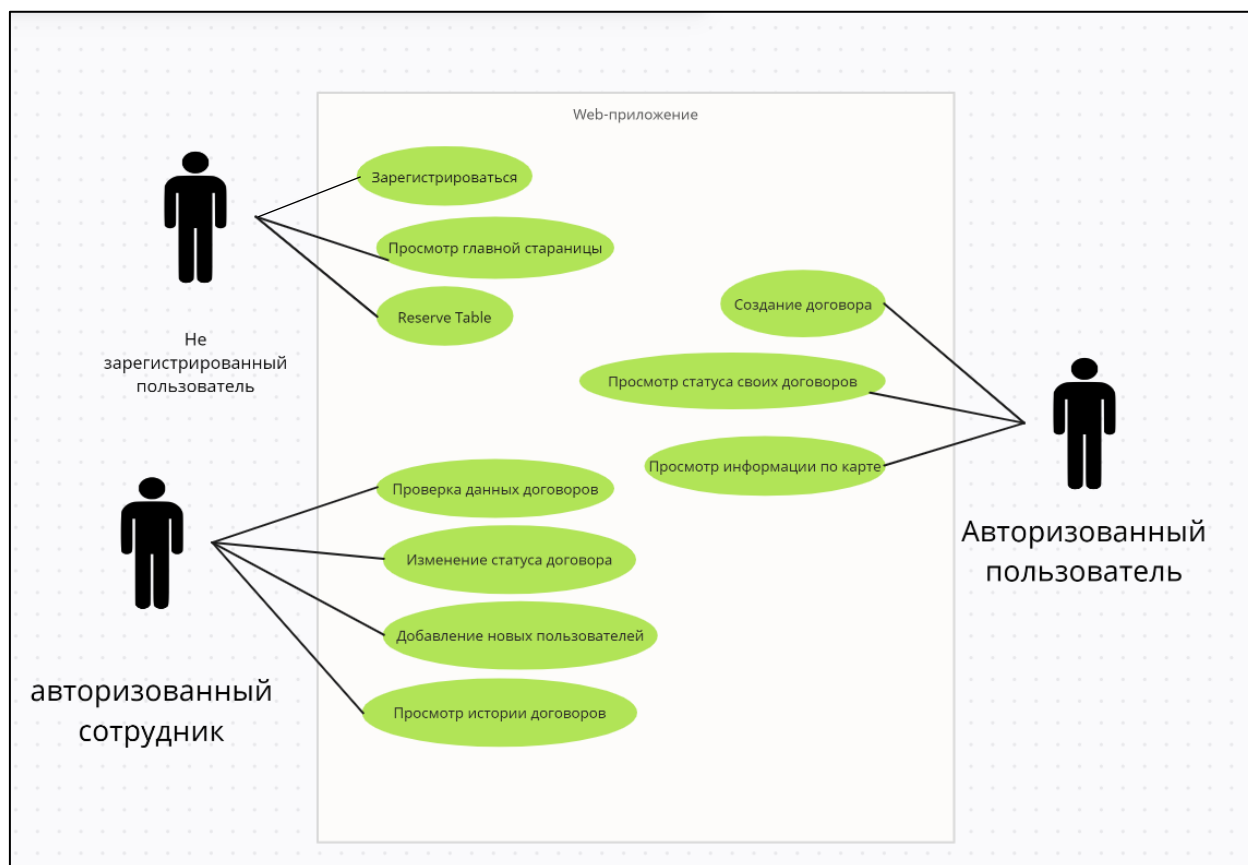


Рис.2. Use-case (функциональная) диаграмма

## 2.4 Физическая модель БД

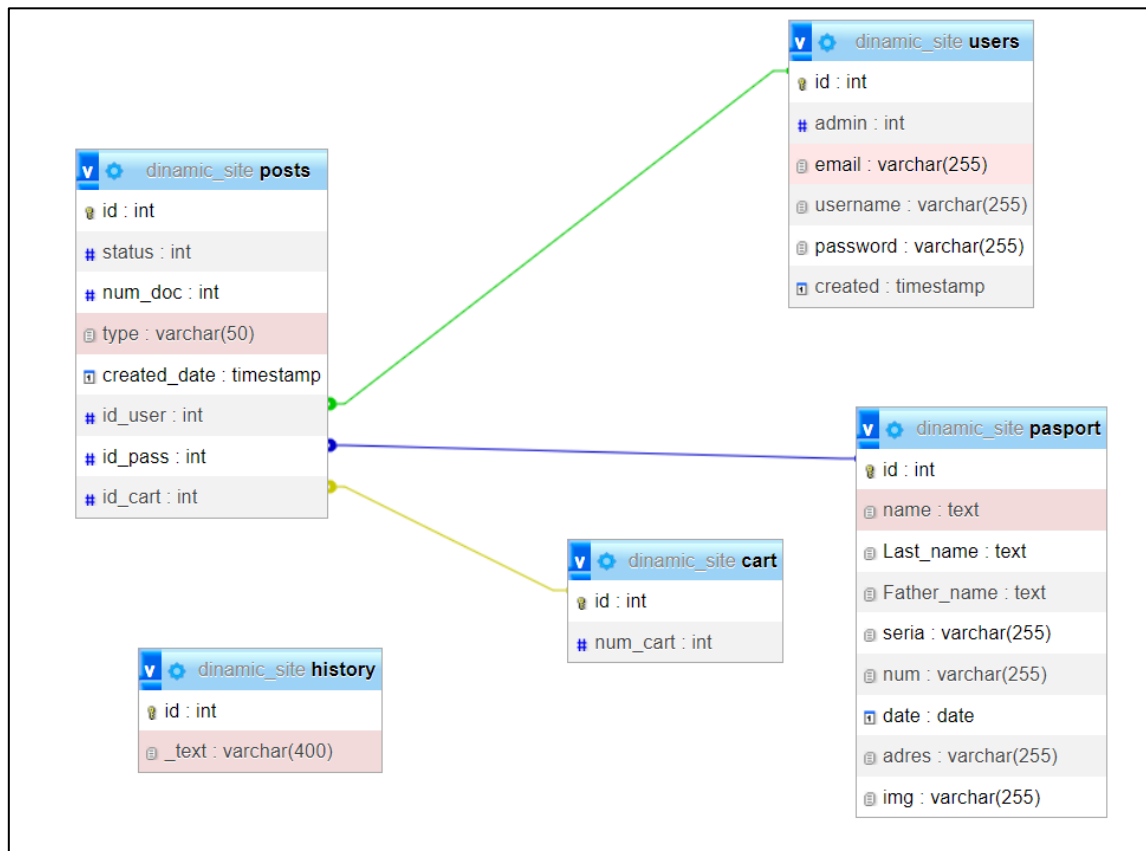


рис. 2.3 Физическая модель БД - нотация IDEF1X

## 3. Реализация

### 3.1 Создание и заполнение базы данных

Скрипт создания таблиц, которой находится в файле `Create_sql.sql`

```

create database if not exists dynamic_site;
use dynamic_site;

create table if not exists users(
    id int not null auto_increment,
    admin int(12) not null,
    email varchar(255) not null,
    username varchar(255) not null,
    password varchar(255) not null,
    created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    primary key(id)
);

create table if not exists passport(
    id int not null auto_increment,
    name text(255) not null,
    Last_name text(255) not null,
    Father_name text(255) not null,
    seria varchar(255) not null,
    num varchar(255) not null,
    date date not null,
    adres varchar(255) not null,
    img varchar(255) not null,
    primary key(id)
);

```

Рис. 3.1

```
create table if not exists cart(  
    id int not null auto_increment,  
    num_cart int(255) not null,  
    primary key(id)  
);  
  
create table if not exists posts(  
    id int not null auto_increment,  
    status int(100) not null,  
    cart_num int(100) not null,  
    type varchar(50) not null,  
    created_date TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    id_user int,  
    id_pass int,  
    id_cart int,  
    foreign key (id_user)  
    references users(id) on delete cascade on update cascade,  
    foreign key (id_pass)  
    references passport(id) on delete cascade on update cascade,  
    foreign key (id_cart)  
    references cart(id) on delete cascade on update cascade,  
    primary key(id)  
);
```

Рис. 3.2

```

create table if not exists history (
    id int not null auto_increment,
    _text varchar(400) not null,
    primary key(id)
);

DELIMITER //
drop trigger if exists pre_insert_posts//
CREATE TRIGGER pre_insert_posts after insert on posts
for each row
    insert into history (_text) values (
        (select concat("Пользователь ",
            (select username from users where id = new.id_user limit 1), " создал договор №", new.id) limit 1)
        );
//

drop trigger if exists pre_update_posts//
CREATE TRIGGER pre_update_posts after update on posts
for each row
    insert into history (_text) values (
        (select concat(
            "У договора №", new.id, " пользователя ",
            (select username from users where id = new.id_user),
            " поменялся статус с ", old.status, " на ", new.status
        )
        )
    );
//
DELIMITER ;

```

Рис. 3.3

На рис. 3.3 также представлены два триггера, позволяющие отслеживать историю изменений в документах.

Скрипт заполнения базы данных:

```

INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('5', '1', 'Maxim_K', 'maxim_375@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('6', '0', 'Greg', 'Greg338811@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('7', '0', 'Tom', 'Tom_0099@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('8', '0', 'Joe', 'Joe_8833@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('9', '0', 'Billie', 'ewewewe@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('10', '0', 'Nikita', 'Nikita02220@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('11', '0', 'Gohha', 'Gohhawwoo22@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('12', '0', 'Nurlan', 'Nurlan3323@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('13', '0', 'Jake', 'Jake20220@mail', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('14', '0', 'Isds99', 'Isds99@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('15', '0', 'Alex2002', 'Alex2002@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('16', '0', 'Nate', 'Nate@mail.ru', '2002-01-01', '2002-01-01');
INSERT INTO `users` (`id`, `admin`, `username`, `email`, `password`, `created`) VALUES ('17', '0', 'Pole', 'Pole@mail.ru', '2002-01-01', '2002-01-01');

INSERT INTO `pasport` (`name`, `Last_name`, `Father_name`, `serial`, `num`, `date`, `img`, `adres`)
VALUES ('Иван', 'Иванов', 'Иванович', '1221', '123321', '2001-01-01', 'invalid.png', 'test');

insert into cart (`num_cart`) values ('1010101');
INSERT INTO `posts` (`id_user`, `status`, `cart_num`, `type`, `created_date`, `id_pass`, `id_cart`)
VALUES ('1', '1', '11223', '1', CURRENT_TIMESTAMP, 1, 1);

```

## 3.2 Описание структуры проекта и используемых технологий

Для реализации веб-приложения был выбран WAMP server.

Для написания backend части приложения был выбран язык PHP(8.0.26). В качестве СУБД использовался MySQL(8.0.31).

Для визуальной части сайта был использован HTML5, CSS, JavaScript.

В главной директории расположены файлы страниц банковских карт (cart.php, cart\_done.php), страница для создания договора (create\_by\_user.php), страница, ответственная за вывод данных договора (get\_all\_my\_posts.php), главная страница (index.php), страницы авторизации, регистрации и logout (log.php, reg.php), файл myDocs, выводящий на страницу данные договора, файл path.php с инициализацией путей, используемых в проекте, файл single.php отвечающий за начальную страницу у авторизованного пользователя.

Также в главной части есть директория admin, в которой находятся директории history, posts и users. Уже в этих папках лежат файлы ответственные за создание/изменение данных сущностей, а также их страницы index.php.

Далее в главной категории находится директория app в которой лежат папки: controllers – здесь файлы posts.php и users.php отвечают за сохранение в бд данных и пользователей и их счетов, а также они содержат код обновления и вывода этих данных.

Database – здесь находится файл подключения к бд (connect.php) и файл с необходимыми функциями для работы с бд (db.php).

Help – файлы отвечают за вывод сообщений с ошибками

Include – включает в себя реализацию хедера для страницы сотрудника, обычного пользователя, бокового меню, а также файл предупреждающий об окончании сессии по истечению 2-х минут.

Далее находится директория assets в которой лежат файлы CSS для работы со стилями, а также здесь находится папка с картинками, которые загружает пользователь при создании договора.

И наконец, в директории sql лежит скрипт создания и заполнения базы данных.

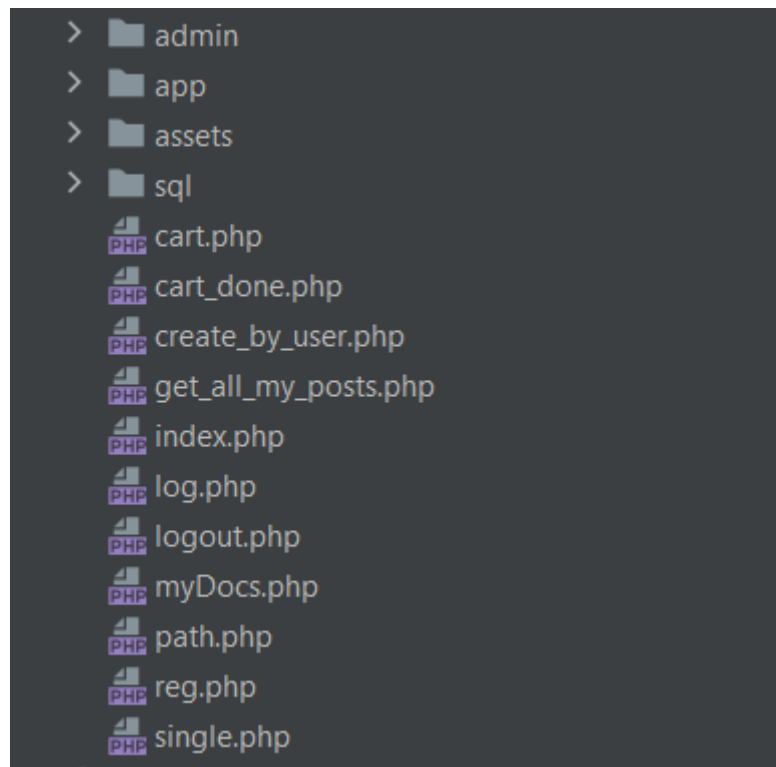


Рис. 3.4 Главная директория

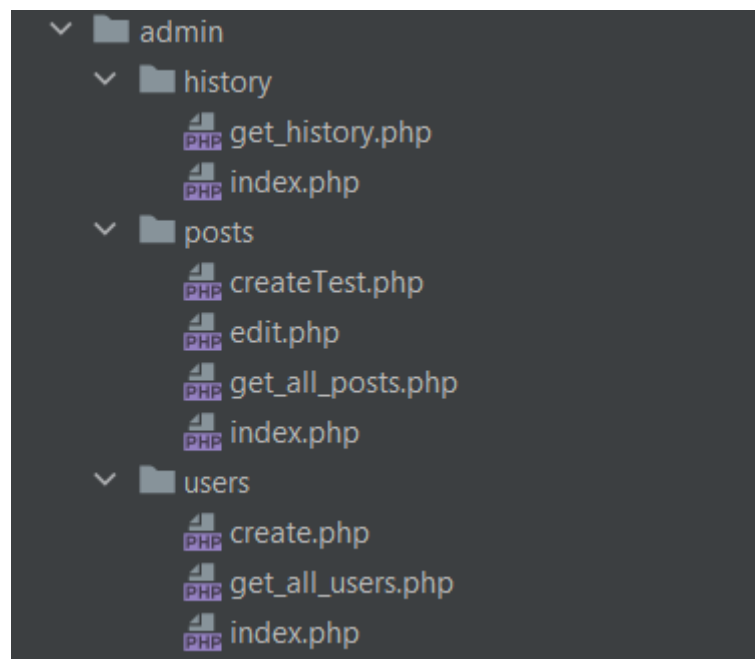


Рис. 3.5 Директория admin



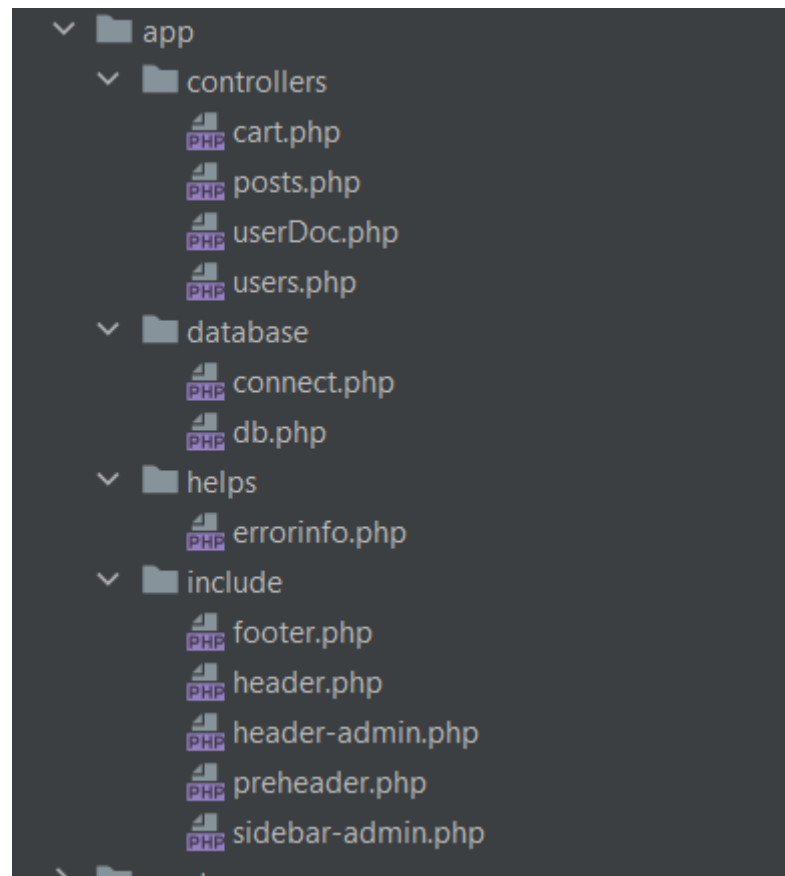


Рис. 3.6 Директория app

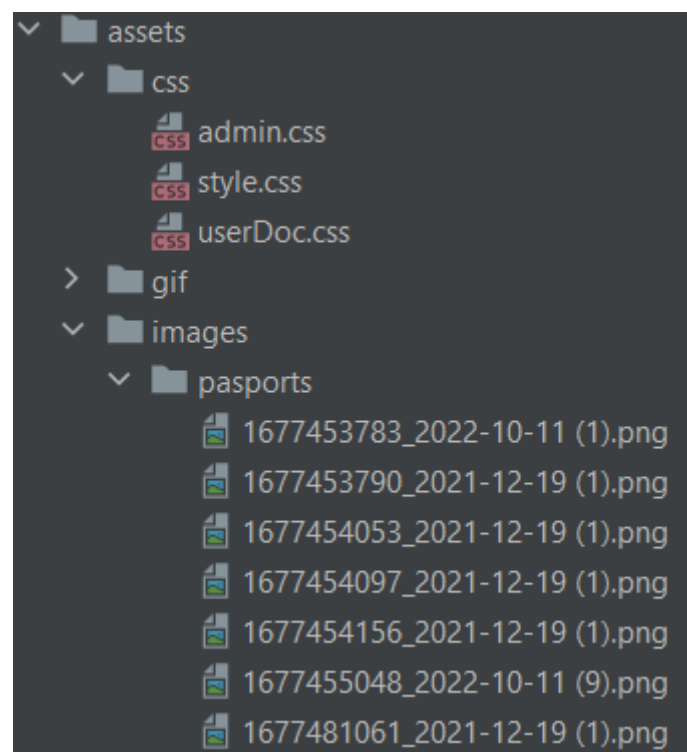


Рис. 3.7 Директория assets

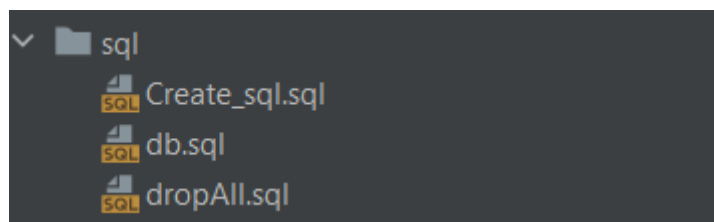


Рис. 3.8 Директория sql

### 3.3 Соответствие требованиям

1. Реализован набор динамических страниц с использованием html, css.
2. Создана база данных dynamic\_site с более чем двумя таблицами.
3. Реализована home страница с возможностями входа и регистрации.
4. Создано 3 типа пользователей с разными правами доступа – неавторизованный пользователь, авторизованный пользователь, сотрудник.
5. Были реализованы страницы с вводом данных в одну или более смежных таблиц.
6. Реализован AJAX запрос, для чего использовался Java Script и JQuery.
7. Были реализованы триггеры для регистрации добавлений и изменений статуса договора.
8. Были реализованы двух минутные сессии для всех пользователей, после чего пользователя переводят на начальную страницу. Реализовано с помощью сессии php и записи cookie файлов.
9. Реализована запись информации о пользователях между сеансами с помощью cookie.
10. Использован метод GET для получения информации об id пользователя.
11. Использован метод POST для создания договора.
12. Реализован параллельный доступ.
13. Страницы соответствуют нормативам Web Content Accessibility Guidelines (WCAG).

## 4. Демонстрация

### 4.1 Демонстрация функционала незарегистрированного пользователя.

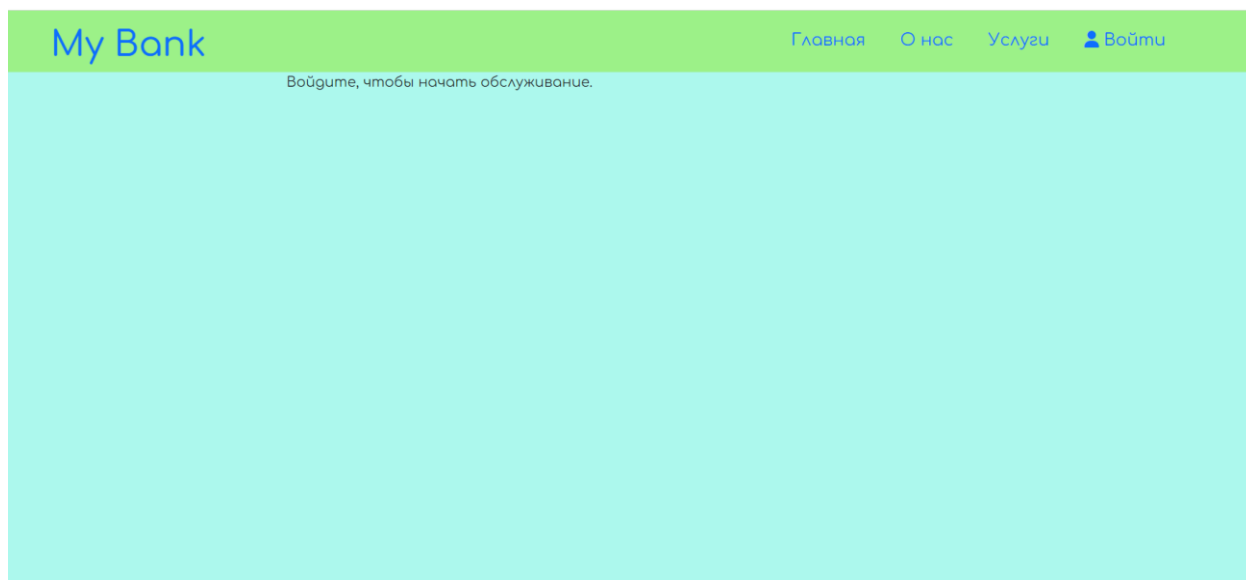


Рис. 4.1 Home страница

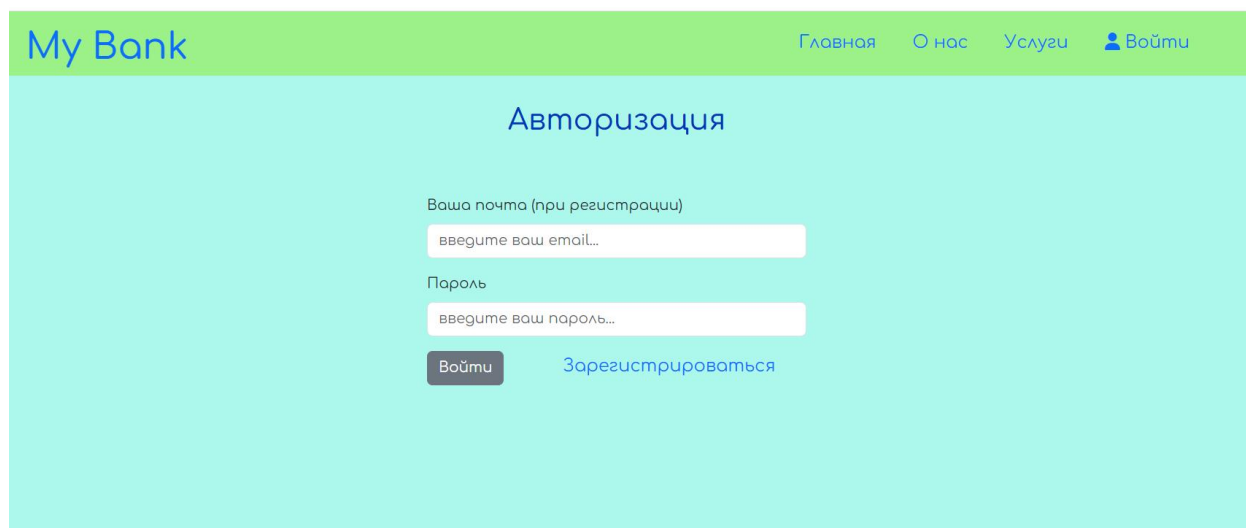


Рис. 4.2 Страница login

My Bank

Главная О нас Услуги Войти

### Форма регистрации

Ваш логин

Email

Ваш email адрес не будет использован для спама

Пароль

Повторите пароль

Регистрация Авторизоваться

Рис. 4.2 Страница reg

Демонстрация проверки вводимых клиентом данных, устранение некорректных данных проверкой на сервере:

Если при авторизации в поле email ввести пробелы вначале строки или в конце, то проверка на сервере уберет эти пробелы и авторизация пройдет успешно.

### Авторизация

Ваша почта (при регистрации)

Пароль

Войти Зарегистрироваться

Рис. 4.3 – пробелы в форме авторизации

```

if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['button-log'])) {
    $email = trim($_POST['email']);
    $pass = trim($_POST['password']);

    if ($email === '' || $pass === '') { //Есть ли пустые поля
        array_push( &array: $errMsg, values: "Не все поля заполнены!");
    }else {
        $existence = selectOne( table: 'users', ['email' => $email]);
        if($existence && password_verify($pass, $existence['password'])) {
            userAuth($existence);
        }else{
            array_push( &array: $errMsg, values: "Почта либо пароль введены неверно");
        }
    }
}

```

Рис. 4.4 – код устранения пробелов

Функция trim убирает ненужные пробелы из формы.

Также здесь демонстрируется использование метода POST при входе.

Примеры попытки регистрации с помощью существующей почты и защиты невалидных данных со стороны сервера в текущей реализации также является примером GET запроса. С его помощью пользователь получает сообщение от сервера, конкретизирующее ошибку.

Рис. 4.5 – проверка существующей почты

## 4.2 Демонстрация функционала клиента

Демонстрация будет проходить на примере пользователя с почтой [aaa@mail.ru](mailto:aaa@mail.ru).

На рисунке 4.6 показано, что у авторизованного пользователя появляются несколько опций: заполнение договора и просмотр уже существующих.

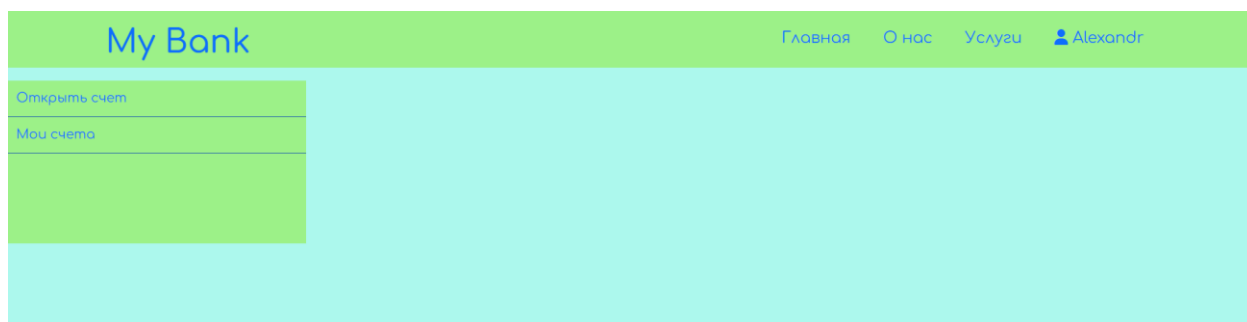


Рис. 4.6

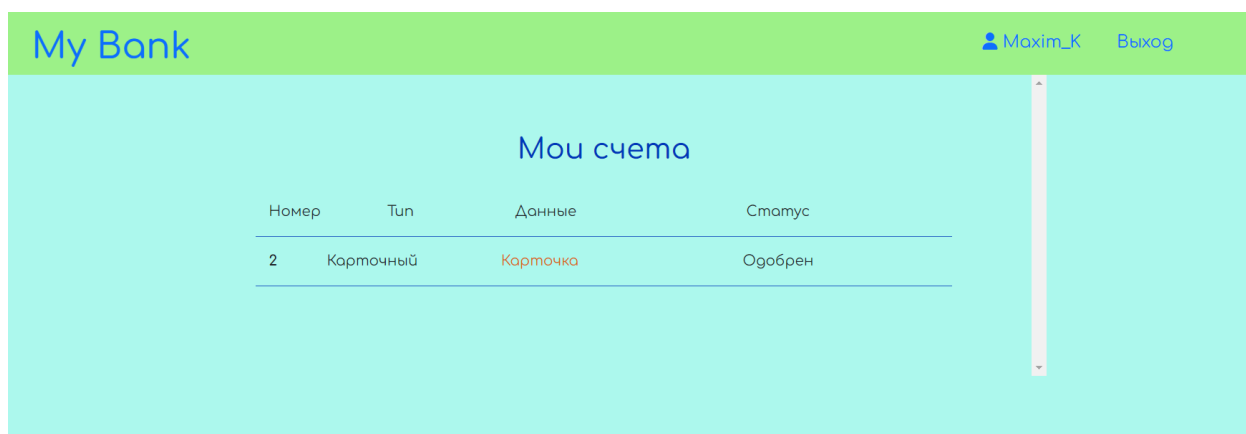


Рис. 4.6.1

Переход от страницы Рис. 4.6.1 к странице Рис. 4.6.2 осуществляется по ссылке “Карточка” на Рис. 4.6.1.

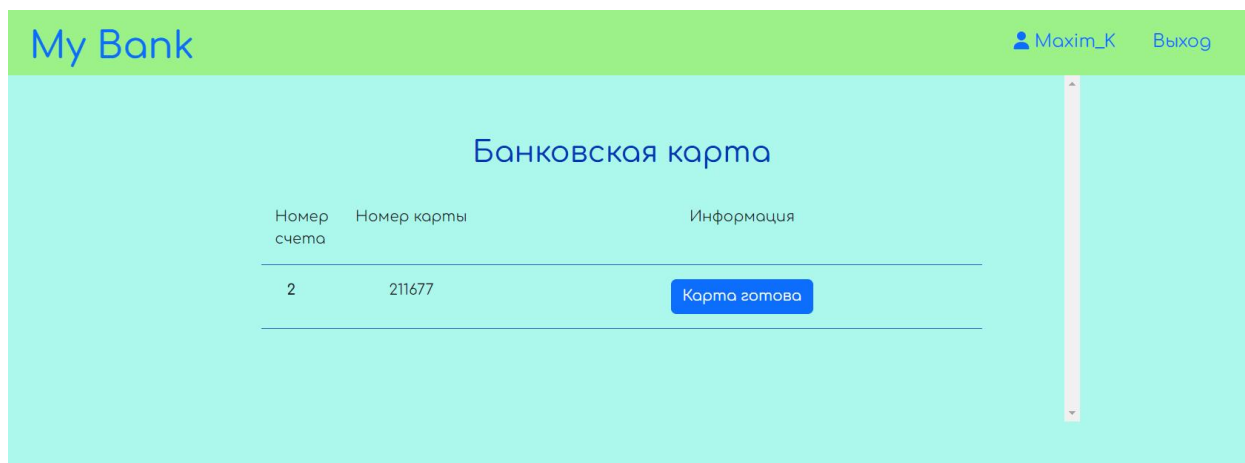


Рис. 4.6.2

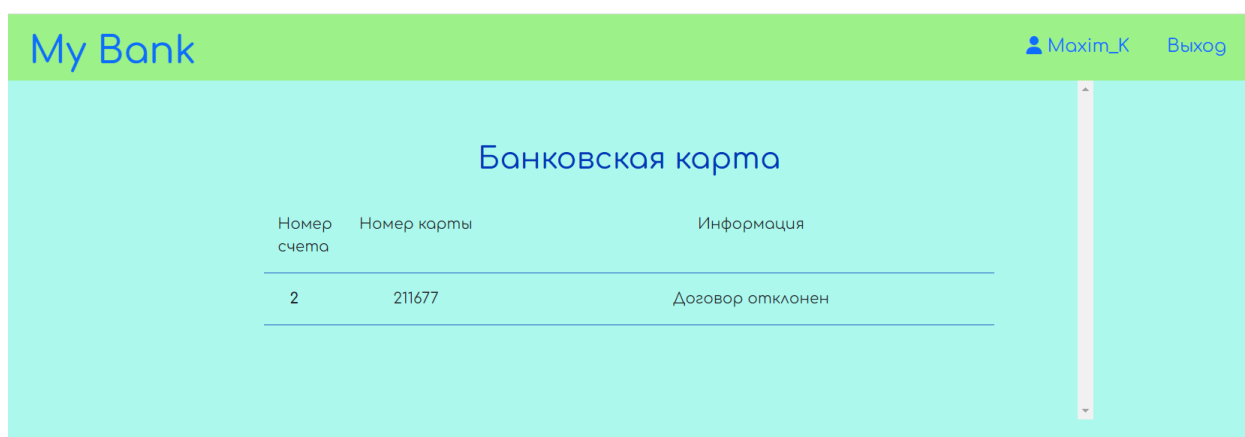


Рис. 4.6.3 – страница с информацией о карта, в случае если договор отклонён

На рис. 4.8 продемонстрирована страница заполнения договора. Не ней реализуются следующие пункты:

Запись данных в три связанные таблицы (posts, cart, passport)

```

$post = [
    'id_user' => $_SESSION['id'],
    'status' => 1, // 1 будет значить что договор сформирован, но не проверен
    'type' => $select,
    'cart_num' => $num_cart,
    'id_pass' => "(select id from passport where seria={$seria} and num={$num})",
    'id_cart' => "(select id from cart where num_cart={$num_cart})"
];

$passport = [
    'name' => $name,
    'last_name' => $l_name,
    'father_name' => $f_name,
    'seria' => $seria,
    'num' => $num,
    'date' => $n_date,
    'adres' => $adres,
    'img' => $img
];

$cart = [
    'num_cart' => $num_cart
];

$passport = insert( table: 'passport', $passport);
$cart = insert( table: 'cart', $cart);
$query = $pdo->prepare( query: "insert into posts (id_user, status, type, cart_num, id_pass, id_cart) values
    ({$$_SESSION['id']}, '1', '{$select}', '{$num_cart}',
    {$passport},
    {$cart}
    );");
$query->execute();
dbCheckError($query);

```

Рис 4.7

	id	status	num_account	type	created_date	id_user	id_pass	id_cart
<input type="checkbox"/> Изменить  Копировать  Удалить	4	2	511677	Кредитный	2023-02-27 12:11:13	18	4	4

Рис 4.7.1 – Результат записи в таблицу posts

	id	name	Last_name	Father_name	seria	num	date	adres	img
<input type="checkbox"/> Изменить  Копировать  Удалить	4	Александр	Лысов	Валерьевич	4543	234564	2023-02-20	г. Москва	1677489073_274px-Pasport_RF.jpg

Рис 4.7.2 – Результат записи в таблицу passport

	id	num_cart
<input type="checkbox"/> Изменить  Копировать  Удалить	4	2147483647

Рис 4.7.3 – Результат записи в таблицу cart



### Заполнение договора

#### Данные паспорта

#### Тип счета

Загрузите первый разворот паспорта и страницу регистрации в формате PDF

Выберите файл

Файл не выбран

Загрузить

Подтвердить

Рис 4.8

В на этой странице так же можно продемонстрировать обработку ошибок пользователя приложением. Например, появляется ошибка, если пользователь указал серию или номер паспорта не подходящей длины:

## Заполнение договора

Серия должна состоять из 4-х символов

### Данные паспорта

Рис 4.9

Проверки также стоят и на другие поля формы:

```
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['post-create'])){
    if (!empty($_FILES['img']['name'])) {
        $imgName = time() . "-" . $_FILES['img']['name'];
        $fileTmpName = $_FILES['img']['tmp_name'];
        $fileType = $_FILES['img']['type'];
        $destination = ROOT_PATH . "\assets\images\pasports\\" . $imgName;

        if (strpos($fileType, 'image') === false) {
            array_push($serrMsg, "Можно загружать только изображения!");
        } else {
            $result = move_uploaded_file($fileTmpName, $destination);

            if ($result) {
                $_POST['img'] = $imgName;
            } else {
                array_push($serrMsg, "Ошибка загрузки изображения на сервер!");
            }
        }
    } else {
        array_push($serrMsg, "Ошибка получения картинки!");
    }
}
```

Рис 4.10 – проверки на картинку

```
if($name === '' || $l_name === '' || $f_name === '' || $seria === '' || $r_date === '' || $adres === '' || $select === '' || $img === ''){
    array_push($serrMsg, "Не все поля заполнены!");
} elseif (mb_strlen($seria, encoding: 'UTF8') < 4){
    array_push($serrMsg, "Серия должна быть более 4-х символов");
} elseif (mb_strlen($num, encoding: 'UTF8') < 6){
    array_push($serrMsg, "Номер должен состоять из шести цифр");
} else{
    // ...
}
```

Рис 4.11 – проверки на пустые поля

### 4.3 Демонстрация функционала сотрудника

Аккаунты инспекторов отличаются от аккаунтов обычных пользователей наличием должности в БД и расширенными возможностями доступа к данным.

Рассмотрим функционал инспектора на примере аккаунта с логином Maxim\_K (maxim\_375@mail.ru, пароль -2002).

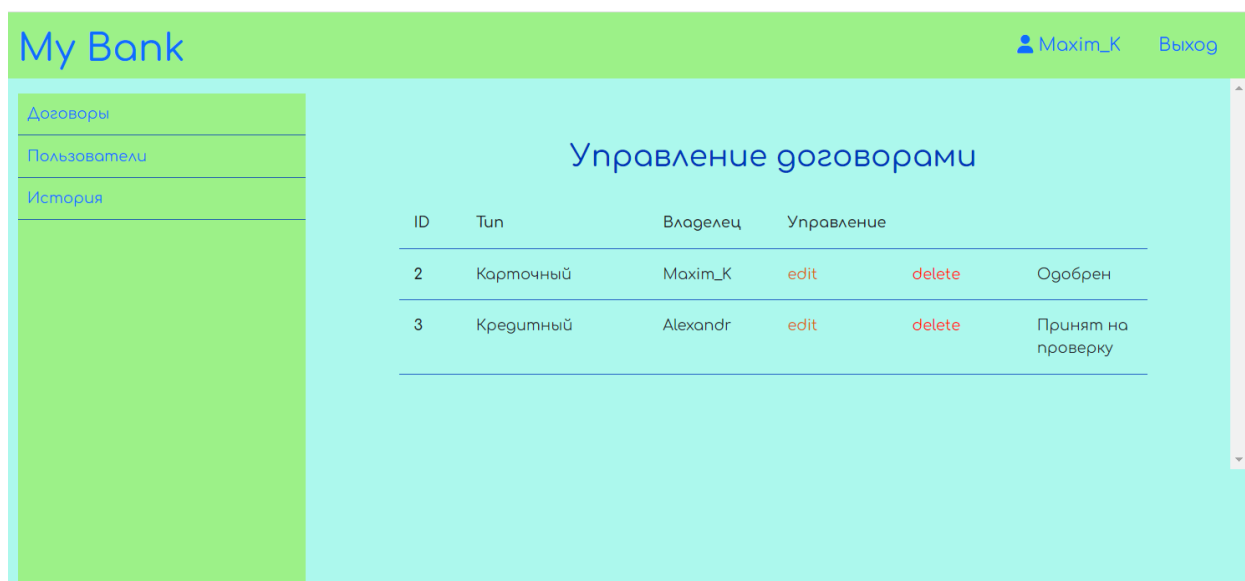


Рис 4.12 – Стартовая страница сотрудника

При нажатии на кнопку edit, сотрудник приступает к проверке договора и попадает на следующую страницу:

Проверка договора

АлександрЛысовВалерьевич

Данные паспорта

4543

234564

2023-02-20

г. Москва

Фото паспорта

РОССИЙСКАЯ ФЕДЕРАЦИЯ

ОТДЕЛОМ ВНУТРЕННИХ ДЕЛ  
ОКТЯБРЬСКОГО ОКРУГА  
ГОРОДА АРХАНГЕЛЬСКА

17.12.2004292-000



Подпись  
Михаил Валерьевич

0000000



ИМЯРЕК

ВЫЕЗЖА

АЛЕКСАНДРОВИЧ

МУЖ.12.09.1982

ГОР. АРХАНГЕЛЬСК

0000000

Подтвердить

Отклонить

Рис 4.13 – Страница проверки договора

На рис. 4.12 представлен вывод данных по разным запросам в поля этой страницы (подтягивается тип счета из posts и с помощью join запроса достается логин из таблицы users).

```
function selectAllFromPostWithUsers($table1, $table2){
    global $pdo;
    $sql = "
    SELECT
    t1.id,
    t1.status,
    t1.type,
    t1.created_date,
    t2.username
    FROM $table1 AS t1 JOIN $table2 AS t2 ON t1.id_user = t2.id
    ";

    $query = $pdo->prepare($sql);
    $query->execute();
    dbCheckError($query);
    return $query ->fetchAll();
}
```

Рис 4.14 – Пример join запроса

```
$postsAdm = selectAllFromPostWithUsers( table1: 'posts', table2: 'users');
```

Рис 4.14.1 использование функции

```
foreach ($postsAdm as $key => $post){ ?>
    <div class="row post">
        <div class="id col-1"><?=$post['id'];?></div>
        <div class="title col-3"><?=$post['type'];?></div>
        <div class="author col-2"><?=$post['username'];?></div>
        <div class="red col-2"><a href="edit.php?id=<?=$post['id'];?>">edit</a></div>
        <div class="del col-2"><a href="edit.php?delete_id=<?=$post['id'];?>">delete</a></div>
        <?php if ($post['status']==1): ?>
            <div class="status col-2">Принят на проверку</div>
        <?php elseif ($post['status']==2): ?>
            <div class="status col-2">Одобен</div>
        <?php elseif ($post['status']==3): ?>
            <div class="status col-2">Отклонён</div>
        <?php elseif ($post['status']==4): ?>
            <div class="status col-2">Уже рассматривается</div>
        <?php endif; ?>
    </div>
<?php } ?>
```

Рис 4.14.2 – вывод данных на страницу (рис. 4.12).

Сотрудник имеет доступ к просмотру истории изменений рис. 4.15. Этот функционал реализован с помощью триггер в базе данных см. рис. 4.16

История изменений	
1	Пользователь Alexandr создал договор №4
2	У договора №1 пользователя Ivan поменялся статус с 1 на 3
3	У договора №2 пользователя Maxim_K поменялся статус с 1 на 2
4	Пользователь Alexandr создал договор №3
5	Пользователь Maxim_K создал договор №2
6	Пользователь Ivan создал договор №1

Рис. 4.15

## 4.4 Использование методов POST и GET

Как говорилось ранее, метод POST используется в форме авторизации (см. рис 4.4), также он используется при создании договора пользователем.

```

if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['post-create'])){
    if (!empty($_FILES['img']['name'])) {
        $imgName = time() . "_" . $_FILES['img']['name'];
        $fileTmpName = $_FILES['img']['tmp_name'];
        $fileType = $_FILES['img']['type'];
        $destination = ROOT_PATH . "\assets\images\pasports\\" . $imgName;

        if (strpos($fileType, 'image') === false) {
            array_push( &array: $errMsg, values: "Можно загружать только изображения!");
        } else {

            $result = move_uploaded_file($fileTmpName, $destination);

            if ($result) {
                $_POST['img'] = $imgName;
            } else {
                array_push( &array: $errMsg, values: "Ошибка загрузки изображения на сервер!");
            }
        }
    }
} else {
    array_push( &array: $errMsg, values: "Ошибка получения картинки!");
}

```

Рис. 4.15.1 – метод post в создании договора.

Метод GET используется для получения id одного договора и паспорта:

```

if($_SERVER['REQUEST_METHOD'] === 'GET' && isset($_GET['id'])){
    //$id = $_GET['id'];
    $post = selectOne( table: 'posts', ['id' => $_GET['id']]);
    $passport = selectOne( table: 'passport', ['id' => $post['id_pass']]);
}

```

Рис. 4.15.2 – метод get.

## 4.5 Реализация триггера

```
DELIMITER //
drop trigger if exists pre_insert_posts//
CREATE TRIGGER pre_insert_posts after insert on posts
for each row
    insert into history (_text) values (
        (select concat("Пользователь ",
            (select username from users where id = new.id_user limit 1), " создал договор №", new.id) limit 1)
        );
//

drop trigger if exists pre_update_posts//
CREATE TRIGGER pre_update_posts after update on posts
for each row
    insert into history (_text) values (
        (select concat(
            "У договора №", new.id, " пользователя ",
            (select username from users where id = new.id_user),
            " поменялся статус с ", old.status, " на ", new.status
        )
        )
    );
//
DELIMITER ;
```

Рис. 4.16

Триггер установлен на отслеживание добавлений договоров и на изменение их статуса. Результат его работы можно увидеть в панели сотрудника(см. рис. 4.15)

При создании пользователем договора его статус равен 1 его значит, что договор принят в обработку. Если договор одобряется после проверки, то его статус меняется на, если отклоняется то на 3.



## 4.6 AJAX запросы

Реализация AJAX запросов присутствует в файле `posts/index.php`, данный запрос (рис. 4.17) позволяет выводить данные на стартовую страницу сотрудника (см. рис 4.12).

```
<script type="text/javascript">
$(document).ready(function(){
    $.ajax({
        url: "get_all_posts.php",
        method: "POST",
        data: {is_admin: true},
        success: function(data){
            $("#preload").remove();
            $("#data_res").html(data);
        }
    });
});
```

Рис. 4.17

## 4.7 Реализация одновременной работы пользователей

Демонстрация параллельного доступа на примере обработки договора одного пользователя двумя разными сотрудниками.

Рассмотрим ситуацию: первый сотрудник по кнопке “edit” заходит на страницу проверки договора пользователя Ivan. И в тоже самое время когда первый сотрудник нажимает кнопку “подтвердить” на странице проверки, второй сотрудник заходит по кнопке “edit” в договор пользователя Ivan. При отсутствии блокировок второй сотрудник приступит к проверке уже проверенного договора. Функционал системы сделан так, что если статус договора не “Принят на проверку”, то при его просмотре нет кнопок “подтвердить” и “отменить”. Получается второй сотрудник увидит эти кнопки и сможет изменить статус повторно.

Для решения этой проблемы используется следующий код:

```

}
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit_post'])){

    /// Таблицы заблокированы, ошибок нет
    lockTables();
    $post = [
        'status' => 2,
    ];
    $id = $_POST['id'];
    sleep( seconds: 10);
    $post_id = update( table: 'posts', $id, $post);
    unlockTables();
    header( header: 'location: ' . BASE_URL . 'admin/posts/index.php');
}
if($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['reject_post'])){
    /// Ошибка при параллельном доступе
    sleep( seconds: 10);
    $post = [
        'status' => 3,
    ];
    $id = $_POST['id'];
    $post_id = update( table: 'posts', $id, $post);
    header( header: 'location: ' . BASE_URL . 'admin/posts/index.php');
}
}

```

Рис. 4.17

Здесь при подтверждении первым сотрудником договора, таблица на запись статуса блокируется на некоторое время, что не позволяет второму пользователю одновременно с проверкой зайти в договор. При отклонении договора намеренно допущена ошибка для демонстрации.

При подтверждении договора первым сотрудником и одновременном переходе туда второго мы не видим кнопок, позволяющих изменить статус (рис. 4.18) – ошибки нет. Но при отклонении договора первым сотрудником, второй еще сможет повлиять на его статус (рис 4.19)

2023-02-20

г. Москва

Фото паспорта



Рис. 4.18

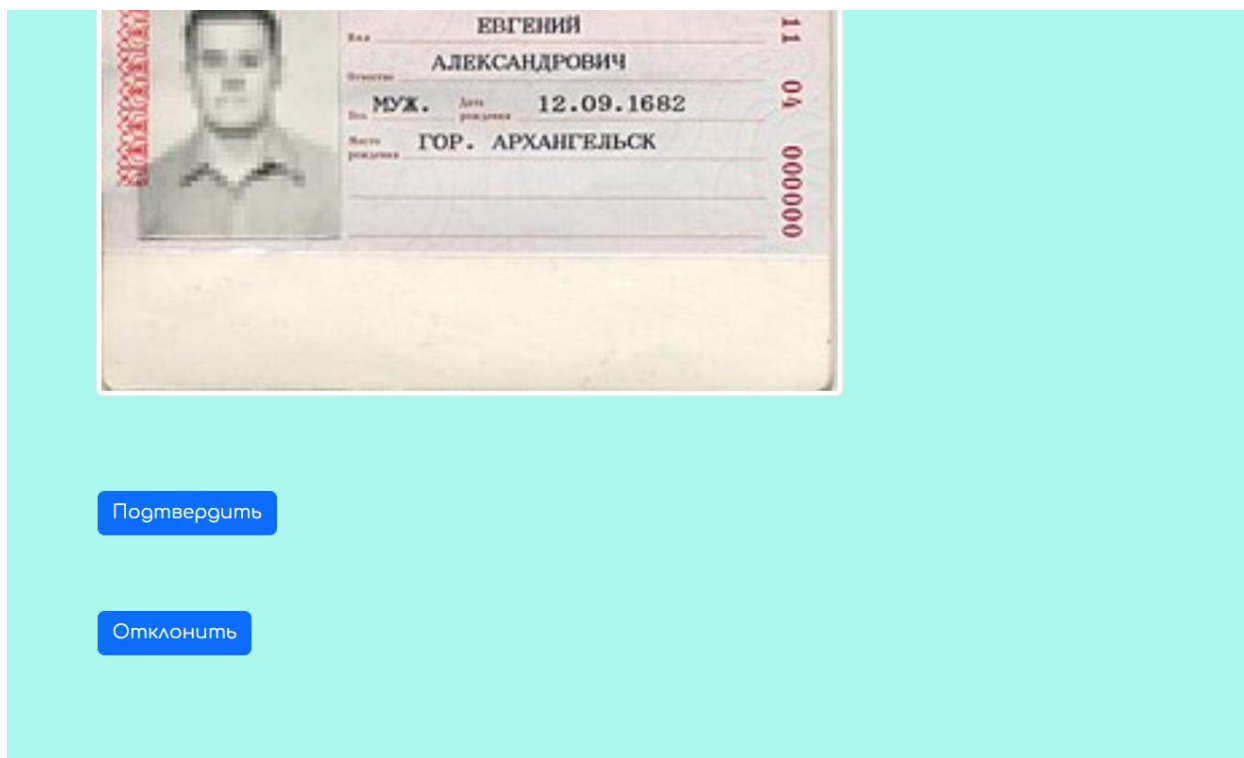


Рис. 4.19

## 4.8 Демонстрация изменений на стороне клиента

Рассмотрим страницу авторизации (рис. 4.3 стр. 15). Введем в форму некорректные данные: почту user1111 и пароль – 1122211122, нажмем “Войти”, при этом появится сообщение об ошибке.

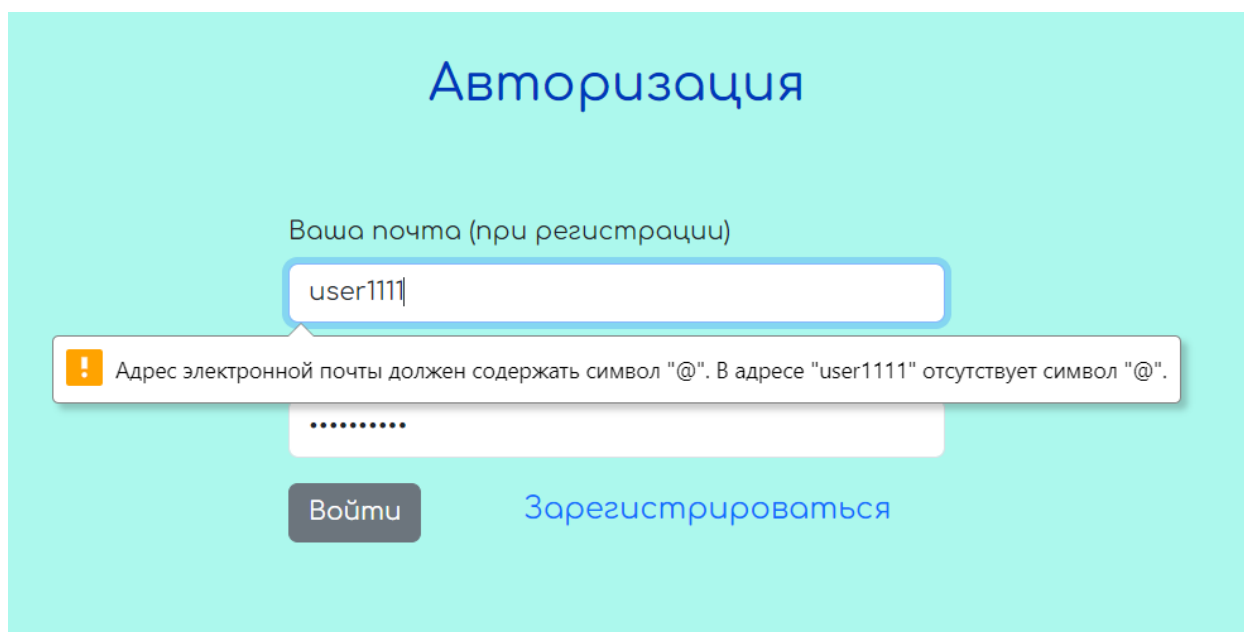


Рис. 4.19.1

Далее нажмем на элемент поле email, запустим правой кнопкой мыши режим разработчика. В строке html кода, ответственного за проверку данных на стороне клиента удалим pattern и поменяем тип поля на text

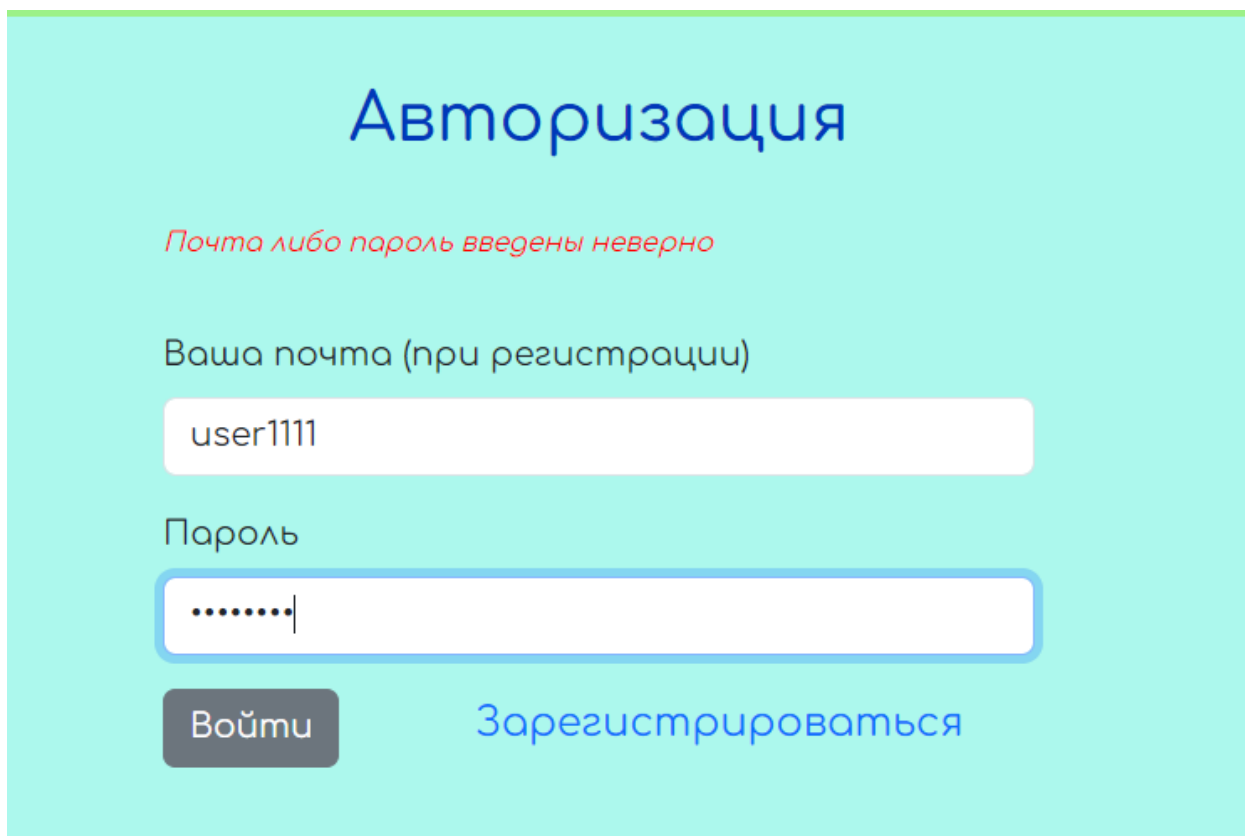
```
<input name="email" value pattern=".+@.+\..+" type="email"
ведите ваш email..."> == $0
```

Рис. 4.19.1

```
<input name="email" value type="text" class="form-control" i
```

Рис. 4.19.2

Попробуем еще раз нажать “Войти”. Получим ошибку, которая пришла со стороны сервера:



Авторизация

*Почта либо пароль введены неверно*

Ваша почта (при регистрации)

user1111

Пароль

.....|

Войти Зарегистрироваться

Рис. 4.19.3

## 4.9 Реализация некоторых требований, которые не были указаны в демонстрации

На рис 4.20 представлена реализация кода, возвращающего пользователя на

начальную страницу по истечению 2-х минут.

```
if(session_status() !== PHP_SESSION_ACTIVE) session_start();
if (isset($_SESSION['LAST_ACTIVITY']) && (time() - $_SESSION['LAST_ACTIVITY'] > 120)) {
    session_unset();
    session_destroy();
    echo "<script type='text/javascript'>alert('Время сессии вышло' );
        window.location.replace('_ . BASE_URL . "/index.php');
    </script>";
}
$_SESSION['LAST_ACTIVITY'] = time(); // update last activity time stamp
```

Рис. 4.20

Функция устанавливающая cookie файлы при регистрации или авторизации пользователя:

```
function userAuth($user){
    $_SESSION['id'] = $user['id'];
    $_SESSION['login'] = $user['username'];
    $_SESSION['admin'] = $user['admin'];
    $_SESSION['email'] = $user['email'];
    $_SESSION['psw'] = $user['password'];
    setcookie('user_id', $user['id'], time() + 1200);
    setcookie('login', $_SESSION['login'], time() + 1200);
    setcookie('email', $user['email'], time() + 1200);
    setcookie('psw', $user['password'], time() + 1200);
    setcookie('admin', $user['admin'], time() + 1200);
    if($_SESSION['admin']){
        header( header: 'location: ' . BASE_URL . "admin/posts/index.php");
    }else{
        header( header: 'location: ' . BASE_URL . "single.php");
    }
}
```

Рис. 4.21

## 4.10 Описание решений по интерфейсу

Выбор цветовой палитры сайта (цветовая палитра – набор оттенков которые используются в проекте) основан на данных из источника [5], где утверждается что цветовая палитра (рис. 5) выглядит гармонично.



Рис. 5 – цветовая палитра.

В рассматриваемом прототипе используются близкие к данной цветовой палитре оттенки, которые так же являются гармоничными согласно рис.6.

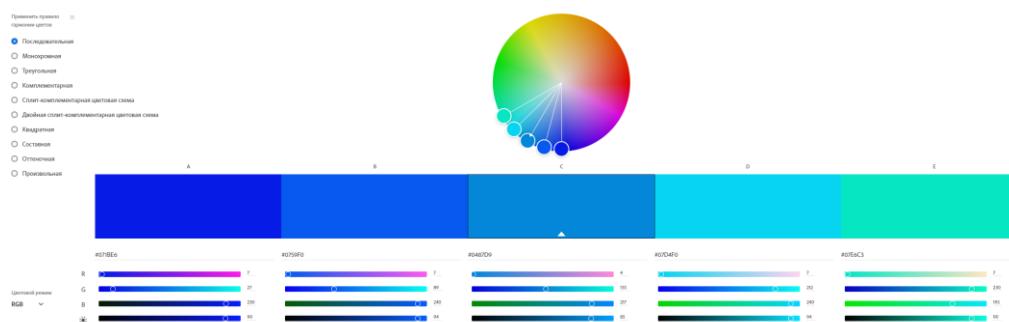


Рис. 6 – анализ гармоничности используемых оттенков.

Основное решение по выбору тускло-бирюзового цвета принято из общих утверждений о том, что фон должен быть не слишком ярк и гармоничен и остальными элементами. Заголовки на сайте покрашены в синий цвет, это сделано из соображений о том, что текст не должен сливаться с фоном, и о том что он должен гармонизировать с фоном сайта. Чёрный текст в таблицах на страницах сотрудников и в таблицах пользователь выбран из-за его контрастности с фоном и его отличии от цвета шрифта на кнопках, это позволяет пользователю интуитивно различать элементы управления и поля со статическими для него данными.

Сообщения о любых ошибках пользователя имеют красный цвет, который показывает неверное использование сайта. В панели сотрудника (см. рис. 4.12) кнопка “edit” имеет оранжевый цвет а, “delete” – красный. Это сделано для лучшего интуитивного понимания сотрудником функционала этих кнопок, хотя предполагается, что сотрудникам хорошо известен весь функционал из инструментов.

Кнопка “Карточка” на странице “мои счета” пользователя имеет оранжевый цвет для привлечения внимания клиента, ведь по нажатию на неё пользователь попадает на страницу и информацией о своей банковской карточке. (Рис. 4.6.2).

Форма и расположение бокового меню на рис 4.12 – Стартовая страница сотрудника и Рис. 4.6 – стартовая страница пользователя, выбрана из принципа обеспечения удобной навигации на сайте.



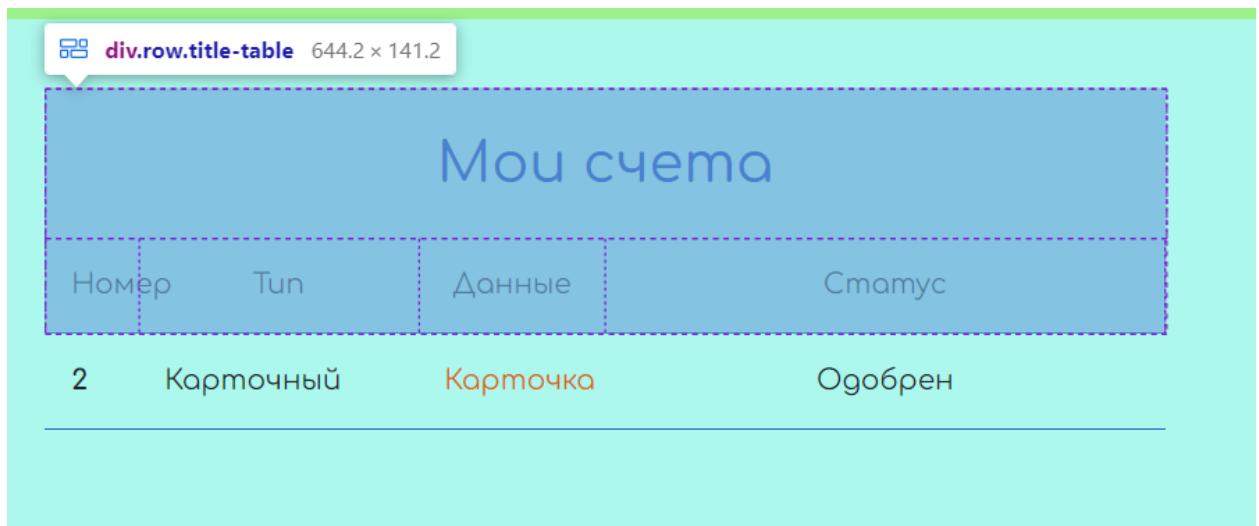


Рис. 4.22 – разделение полей заголовков



Рис. 4.23 – разделение полей данных

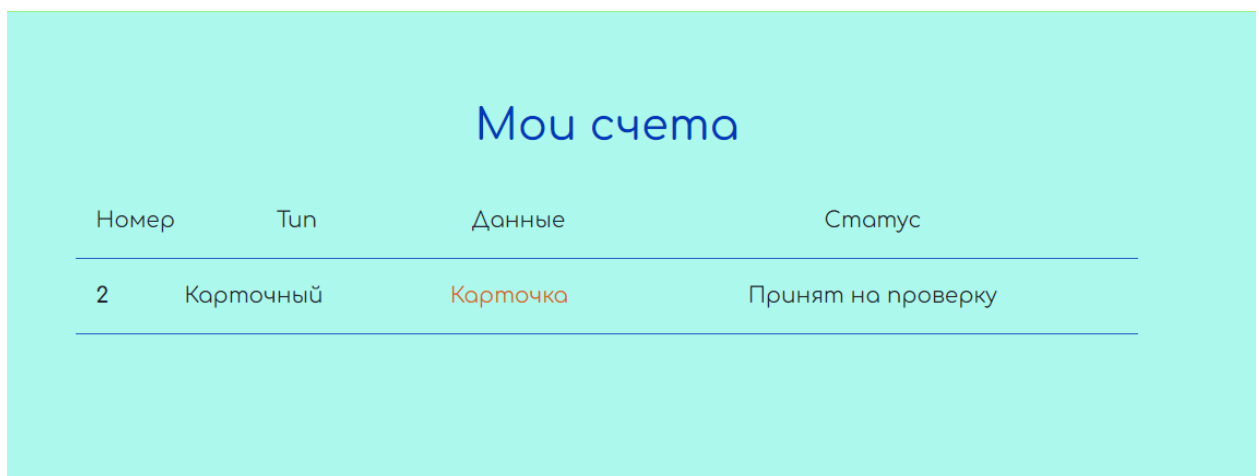
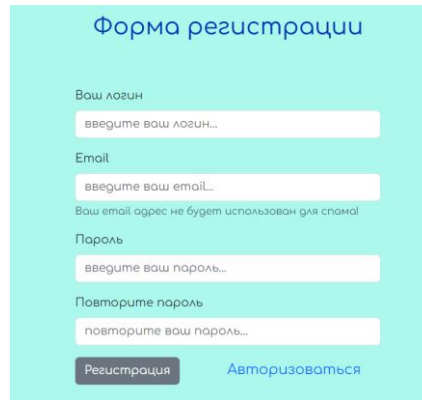


Рис. 4.24 – Общий вид

## 5 Сценарий демонстрации

1. Для демонстрации всех функций работы приложения, необходимо со стартовой страницы перейти на страницу регистрации. (Интерфейс по нормам Web Content Accessibility Guidelines (WCAG)).



Форма регистрации

Ваш логин  
введите ваш логин...

Email  
введите ваш email...  
Ваш email адрес не будет использован для спама!

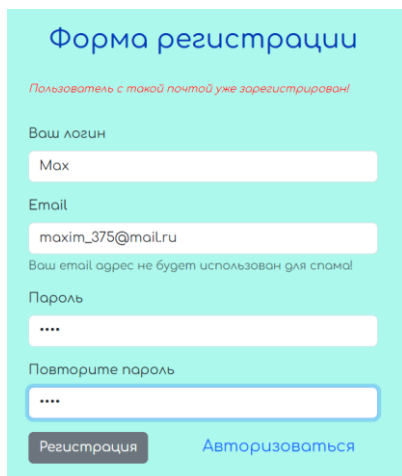
Пароль  
введите ваш пароль...

Повторите пароль  
повторите ваш пароль...

Регистрация Авторизоваться

Рис. 6.1 – страница регистрации.

2. Ввести некорректные символы в поле почты, ввести уже существующую почту - получить ошибку.



Форма регистрации

Пользователь с такой почтой уже зарегистрирован!

Ваш логин  
Max

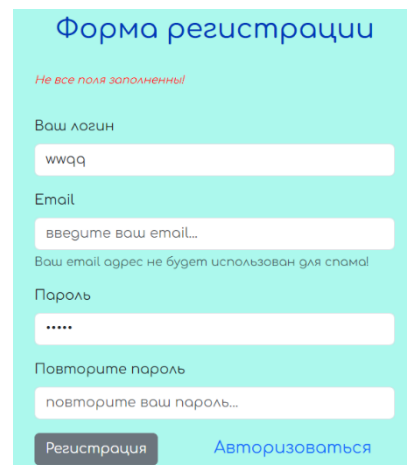
Email  
maxim\_375@mail.ru  
Ваш email адрес не будет использован для спама!

Пароль  
....

Повторите пароль  
....

Регистрация Авторизоваться

Рис. 6.2- ошибка, т.к. пользователь с такой почтой уже зарегистрирован (рис. 6.3)



Форма регистрации

Не все поля заполнены!

Ваш логин  
wwqq

Email  
введите ваш email...  
Ваш email адрес не будет использован для спама!

Пароль  
.....

Повторите пароль  
повторите ваш пароль...

Регистрация Авторизоваться

Рис. 6.3- ошибка, т.к. не все поля заполнены.

□ Изменить Копировать Удалить 5 1 maxim\_375@mail.ru Maxim\_K \$2y\$10\$.YoFxnNKrNGbYHrQUA8lQufePaAuuFdB7D33hjtTn69... 2023-02-27 02:30:20

Рис. 6.4 – наличие пользователя в бд, с помощью почты которого пытаются зарегистрироваться.

3. Зарегистрироваться как пользователь с логином Alex, почтой [alex@mail.ru](mailto:alex@mail.ru) и паролем 1234.

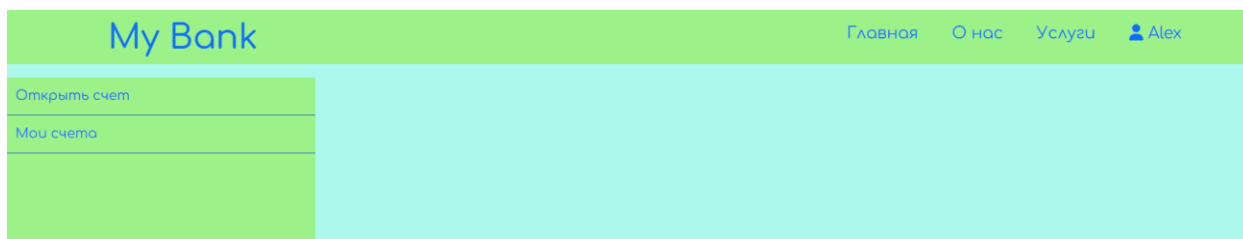


Рис. 6.5 - стартовая страница авторизованного пользователя.

4. Попасть на начальную страницу клиента, затем зайти в “открыть счет”.

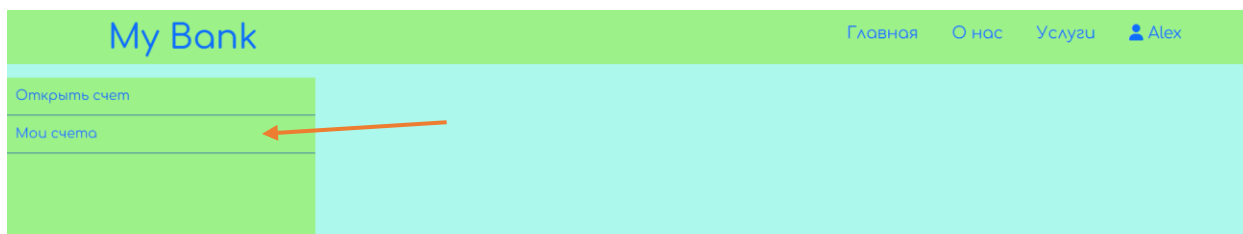


Рис. 6.6 - стартовая страница авторизованного пользователя, переход на страницу “Мои счета”.(далее будут появляться красные стрелки для лучшего понимания действий).

5. Произвести заполнение данных: ввести ФИО Алексей Белов Петрович, серия паспорта - 123, номер паспорта - 121544, дата выдачи - 01.02.2002, Адрес регистрации – г. Москва, тип счета – кредитный, загрузить скан паспорта в формате jpg.

Рис. 6.7 -ввод не корректных данных в договор

6. Получить ошибки: “серия паспорта должна состоять из 4-х символов”.

Заполнение договора

Серия должна состоять из 4-х символов

Алексей Белов Петрович

Данные паспорта

123

Рис. 6.8 – получение ошибки из-за длины серии.

7. Ввести серию 1234, и загрузить скан паспорта в формате doc. Получить ошибку.

Заполнение договора

Можно загружать только изображения!  
Не все поля заполнены!

Алексей Белов Петрович

Данные паспорта

1234

123312

01.02.2002

г. Москва

Тип счета

Рис. 6.8 – получение ошибки из-за формата изображения.

9. Ввести в поле серия значение 1234, а в поле загрузки скана паспорта добавить файл формата jpg или pdf, нажать кнопку “Подтвердить”.

10. Навести курсор на свой логин в правом верхнем углу, в выпадающем меню нажать кнопку “Выход”.

My Bank

Главная О нас Услуги Alex

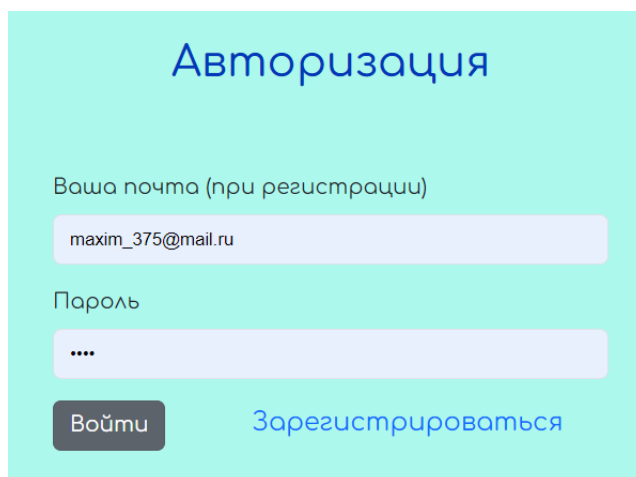
Открыть счет

Мои счета

Выход

Рис. 6.9 – выход из аккаунта.

11. Снова переместить курсор в правый верхний угол, нажать “Войти”. Ввести почту – [maxim\\_375@mail.ru](mailto:maxim_375@mail.ru) и пароль 2002, нажать “Войти”,



Авторизация

Ваша почта (при регистрации)

maxim\_375@mail.ru

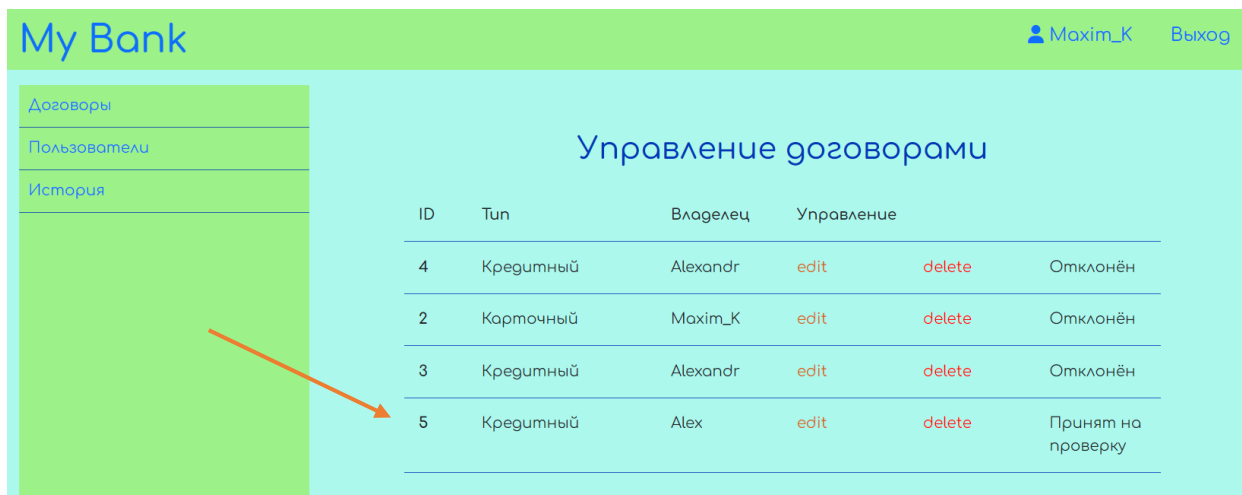
Пароль

....

Войти Зарегистрироваться

Рис. 6.10 – вход в аккаунт.

12. На главной странице сотрудников увидеть корректное отображение созданного ранее договора, убедиться что его статус “Принят на проверку”.



My Bank

Maxim\_K Выход

Договоры  
Пользователи  
История

Управление договорами

ID	Тип	Владелец	Управление
4	Кредитный	Alexandr	edit delete Отклонён
2	Карточный	Maxim_K	edit delete Отклонён
3	Кредитный	Alexandr	edit delete Отклонён
5	Кредитный	Alex	edit delete Принят на проверку

Рис. 6.11 –демонстрация корректного отображения договора.

13. В левом меню нажать кнопку “История”, убедиться что на этой странице появилась запись “Пользователь Alex создал договор №1”, вернуться на страницу с договорами.

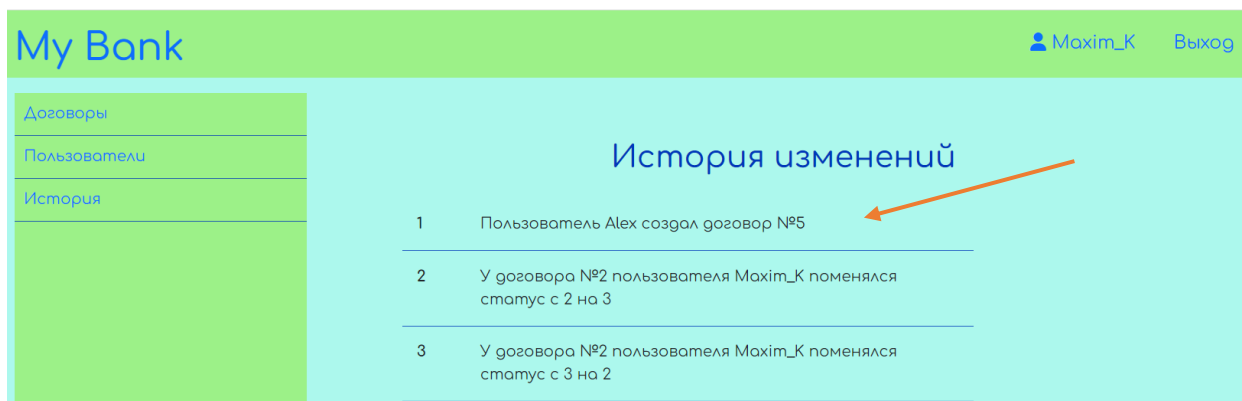


Рис. 6.11 – демонстрация корректной работы триггера.

14. Открыть второй браузер, авторизоваться в нем как сотрудник, т.е. в поле регистрации ввести почту – [adm@mail.ru](mailto:adm@mail.ru) и пароль – 2002. Попасть на главную страницу сотрудников.

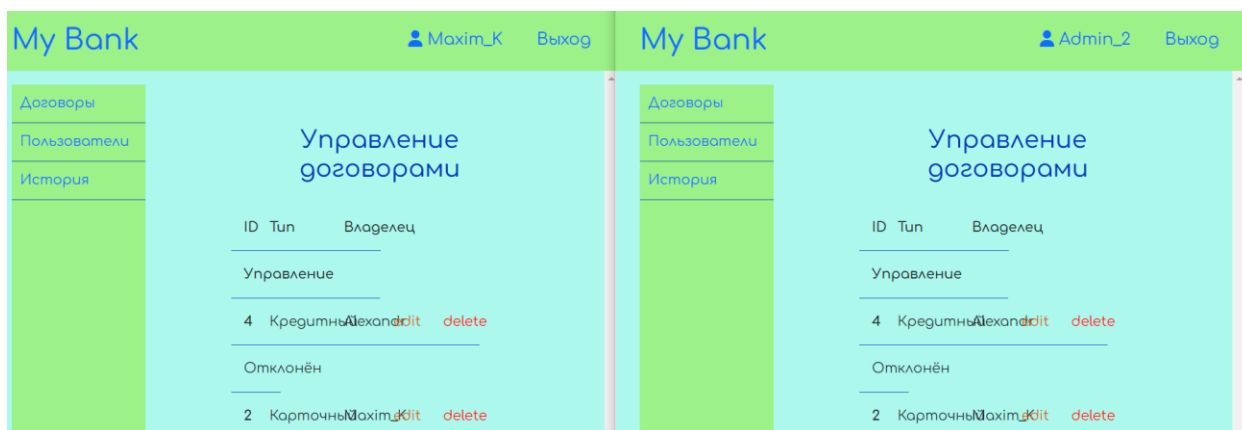


Рис. 6.12 – использование 2-х браузеров.

15. В первом окне (сотрудник Maxim\_K) нажать кнопку “edit” в договоре с id: 5. Проверить корректность данных и нажать кнопку “Подтвердить”. (если нажать “отклонить”, то на шаге 16 убедиться в получении ошибки, второй сотрудник сможет изменить статус договора, но мы демонстрируем корректную работу, т.к. это нужно для дальнейших шагов, описание ошибки параллельного доступа находится в пункте “Реализация параллельного диспута”).

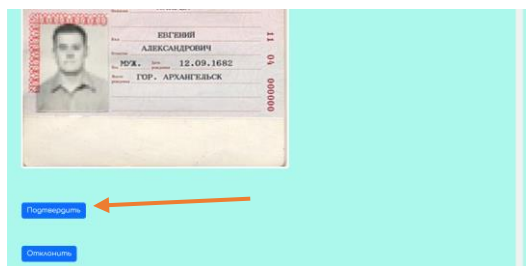


Рис. 6.13 – первый сотрудник подтверждает договор.



Рис 6.14 – у 2-го сотрудника нет кнопок.

16. Переключиться на второго сотрудника (Admin\_2) нажать кнопку “edit”, убедиться, что таблицы заблокированы, статус договора при этом поменялся на 2, а это значит, что кнопки “Подтвердить” и “отклонить” отсутствуют и второй сотрудник не может повторно изменить статус.



Рис 6.14 – у 2-го сотрудника нет кнопок.

17. Закрыть и открыть окно в котором первый сотрудник (Maxim\_K). (cookie между сеансами).

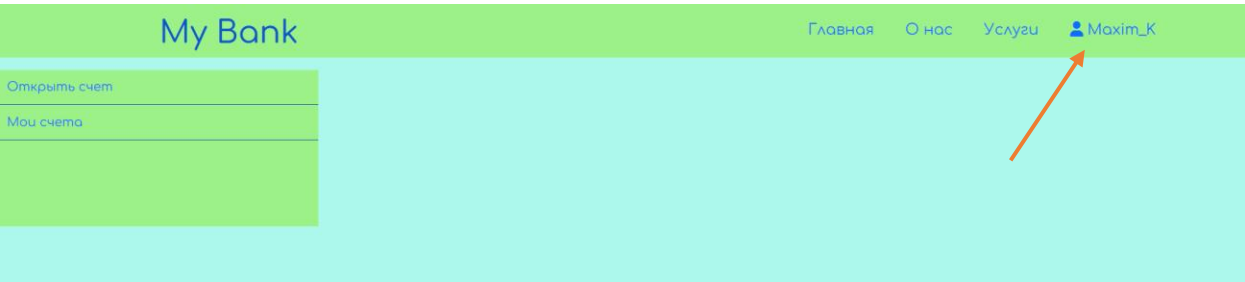


Рис 6.15 – при открытии сайта сотрудник авторизован.

Name	Value	Domain	Path	Expires...	Size	HttpOnly	Secure	SameSi...	SameP...	Partitio...	Priority
__stripe_mid	cefb421f-59e0-4f0c-b222-22f05a796a915e0fc3	.fontaw...	/	2024-0...	54		✓	Strict			Medium
SOFTCookies1405_sid	qh5tneiSdbOI87hPHgkraRvKHp8ulcva	localhost	/	Session	51			Strict			Medium
Phpstorm-9e54db1b	7953ffd4-57de-4ebd-b0d6-d9818f7e7560	localhost	/	2024-0...	53	✓		Strict			Medium
Webstorm-94253fbf	7cee49bd-7ec9-4514-abea-3ed0379bdb1a	localhost	/	2024-0...	53	✓		Strict			Medium
user_id	5	localhost	/dinam...	2023-0...	8						Medium
admin	1	localhost	/dinam...	2023-0...	6						Medium
PHPSESSID	nceoq3q1cv2hkajam9f7nc21f1	localhost	/	Session	35						Medium
psw	%242y%2410%24.YoFxnNKrNGbYHrQUA8IQufeP...	localhost	/dinam...	2023-0...	69						Medium
SOFTCookies6480_sid	T07iwLeHwJ4hYEBvK9uRUgqqxmRjFE48	localhost	/	Session	51						Medium
email	maxim_375%40mail.ru	localhost	/dinam...	2023-0...	24						Medium
login	Maxim_K	localhost	/dinam...	2023-0...	12						Medium

Рис 6.16 – сохранение cookie на сайте.

18. Войти как пользователь с логином Alex, почтой alex@mail.ru и паролем 1234.

20. Зайти в аккаунт Alex (почта alex@mail.ru, пароль 1234) В боковом меню нажать на “Мои счета”. Убедиться что статус договора “Одобен”.



Рис 6.17 – проверка статуса

21. Нажать на ссылку “Карточка”, попасть на страницу с информацией о карте.

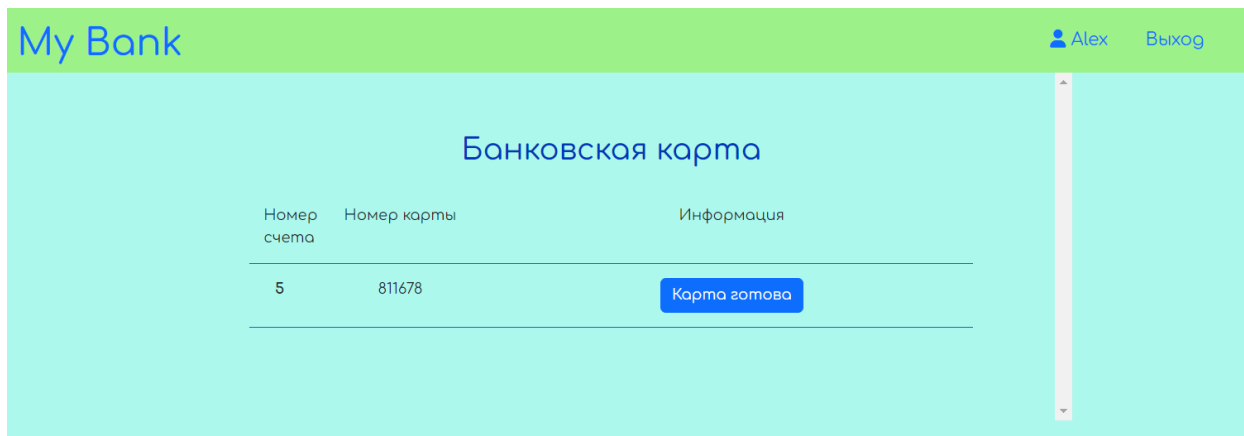


Рис 6.18 – страница с информацией о карте.

22. Нажать кнопку “Карточка”, попасть на страницу с сообщением “Ваша карта готова, заберите её в течении недели”.

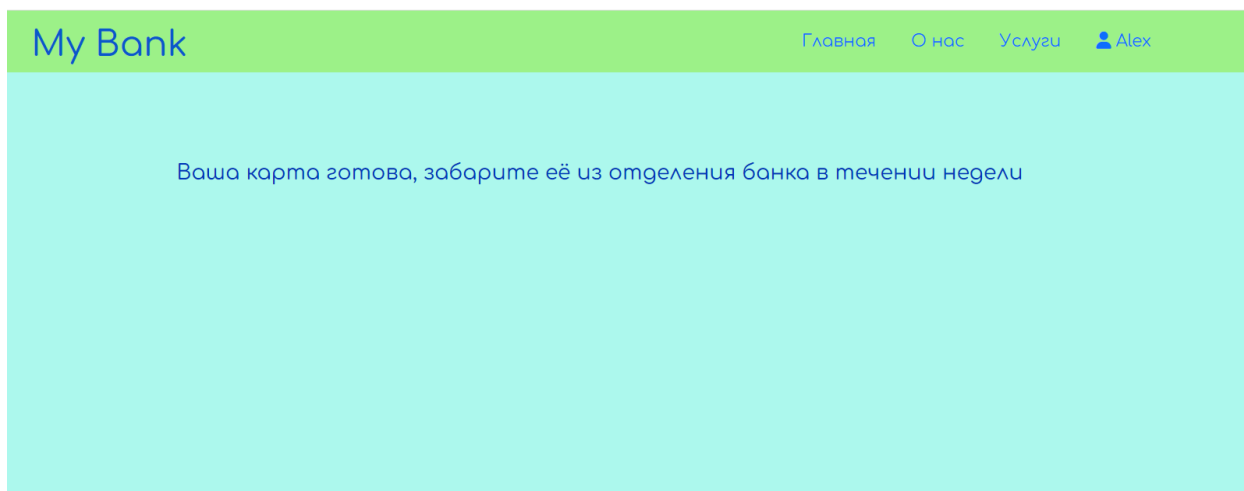
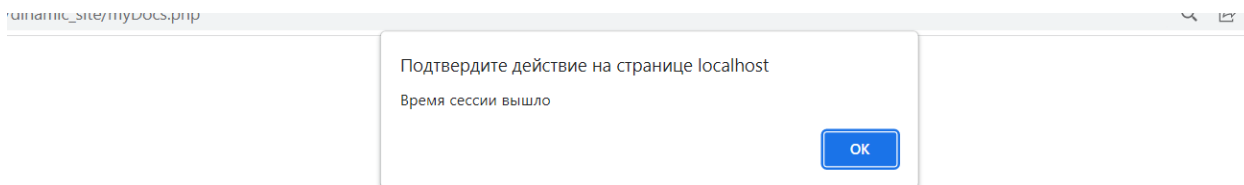


Рис 6.19 – страница с информацией о карте.

23. Подождать 2 минуты, получить сообщением об окончании сессии, автоматически переместиться на главную страницу.





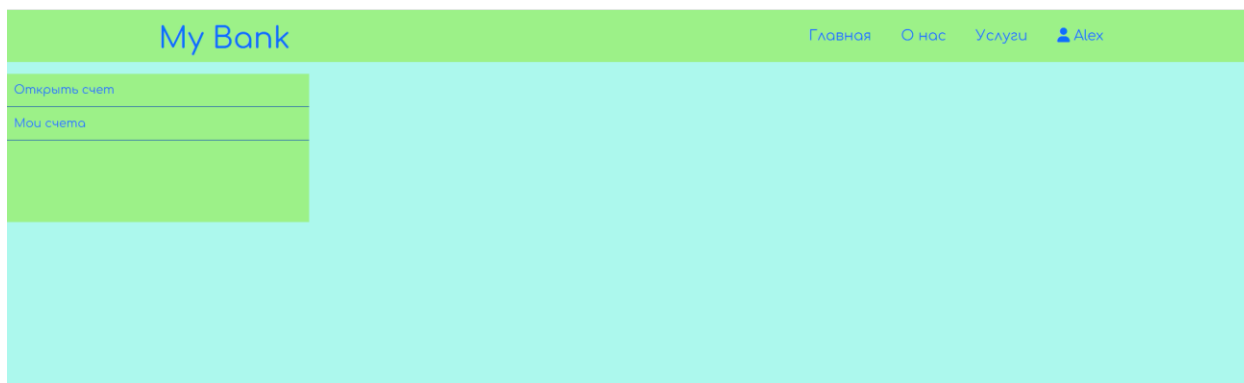


Рис 6.30 – Возвращение на главную страницу при нажатии на ОК.

## 6 Источники

1. <https://www.sravni.ru/banki/info/kak-otkryt-schet-v-sberbanke/>
2. [https://www.raiffeisen.ru/business/rko/online\\_request/](https://www.raiffeisen.ru/business/rko/online_request/)
3. <https://www.monito.com/ru/wiki/otkryt-schet-v-sberbanke>
4. <https://www.vbrr.ru/private/currents/contract-rur.pdf>
5. <https://lpgenerator.ru/blog/2017/06/01/spisok-resursov-dlya-sozdaniya-idealnoj-cvetovoj-palitry-vashego-sajta/>