

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**  
**по дисциплине «Введение в нереляционные базы данных»**  
**Тема: Сервис анализа данных для выдачи кредитов - Профили клиентов,**  
**составление рейтингов, прогнозы**

Студент гр. 1384	_____	Камынин А. А.
Студентка гр. 1384	_____	Усачева Д. В.
Студент гр. 1384	_____	Бобков В. Д.
Преподаватель	_____	Заславский М. М.

Санкт-Петербург  
2024

## ЗАДАНИЕ

Студент Камынин А. А.

Студентка Усачева Д. В.

Студент Бобков В. Д.

Группа 1384

Тема: Сервис анализа данных для выдачи кредитов - Профили клиентов, составление рейтингов, прогнозы

Исходные данные:

Необходимо создать веб-приложение с базой знаний и возможностью выдачи, отслеживания и получения кредитов с использованием СУБД MongoDB.

Содержание пояснительной записки:

«Содержание»

«Введение»

«Сценарии использования»

«Модель данных»

«Разработанное приложение»

«Выводы»

«Приложения»

«Список использованных источников»

Предполагаемый объем пояснительной записки:

Не менее 15 страниц.

Дата выдачи задания: 05.09.2024

Дата сдачи реферата: 22.12.2024

Дата защиты реферата: 24.12.2024

Студент	_____	Камынин А. А.
Студентка	_____	Усачева Д. В.
Студент	_____	Бобков В. Д.
Преподаватель	_____	Заславский М. М.

## **АННОТАЦИЯ**

В рамках ИДЗ было разработано веб-приложение для выдачи, отслеживания и получения кредитов. Приложение включает функционал для взятия, просмотра и принятия решения по кредитам, а также для просмотра профиля и статистики. Реализована система фильтрации, поиска и сортировки для кредитов, заявок, кредитной истории и истории взаимодействия по различным критериям.

В процессе разработки были использованы технологии Vue.js, Flask, СУБД MongoDB и Docker.

Исходный код доступен по ссылке: [nosql2h24-loans](https://github.com/nosql2h24-loans).

## **SUMMARY**

The individual homework assignment developed a web-based application for issuing, tracking and receiving loans. The application includes functionality for taking, reviewing and deciding on loans, as well as for viewing profile and statistics. A filtering, search and sorting system for loans, applications, credit history and interaction history by various criteria was implemented.

Vue.js, Flask, MongoDB and Docker DBMS technologies were used in the development process.

The source code is available at the link: [nosql 2h24-loans](https://github.com/nosql2h24-loans).

## СОДЕРЖАНИЕ

1.	Введение	7
1.1.	Актуальность проблемы	7
1.2.	Постановка задачи	7
1.3.	Предлагаемое решение	7
1.4.	Качественные требования к решению	8
2.	Сценарии использования	9
2.1.	Макет UI	9
2.2.	Сценарий использования: Регистрация	16
2.3.	Сценарий использования: Профиль сотрудника банка	16
2.4.	Сценарий использования: Просмотр заявок на кредиты	17
2.5.	Сценарий использования: Просмотр кредитной истории клиента	18
2.6.	Сценарий использования: История взаимодействия сотрудника с клиентом	20
2.7.	Сценарий использования: Статистика банка	21
2.8.	Сценарий использования: Массовый импорт/экспорт данных	22
2.9.	Сценарий использования: Профиль клиента	23
2.10.	Сценарий использования: Просмотр открытых кредитов	24
2.11.	Сценарий использования: Оформление кредита	26
2.12.	Сценарий использования: Заявки на оформление кредита клиента	27
2.13.	Вывод	28
3.	Модель данных	29
3.1.	Нереляционная модель данных	29
3.2.	Реляционная модель данных	36
3.3.	Сравнение моделей	43
4.	Разработанное приложение	46

4.1.	Краткое описание приложения	46
4.2.	Использованные технологии	46
4.3.	Схема экранов приложения	46
5.	Выводы	48
5.1.	Достигнутые результаты	48
5.2.	Недостатки и пути для улучшения полученного решения	48
5.3.	Будущее развитие решения	49
6.	Приложения	51
6.1.	Документация по сборке и развертыванию приложения	51
6.2.	Инструкция для пользователя	51
	Список использованных источников	54

## **ВВЕДЕНИЕ**

### **1.1. Актуальность проблемы**

С увеличением спроса на цифровые инструменты для управления финансами, включая оформление и отслеживание кредитов, возрастает потребность в удобных и функциональных платформах для работы с кредитными продуктами. Пользователи часто сталкиваются с проблемами недостаточной прозрачности кредитных условий, сложностями в управлении заявками и отслеживании кредитной истории. Существующие решения нередко имеют ограничения в функционале фильтрации, поиска и анализа данных, а также не обеспечивают должного уровня пользовательского взаимодействия.

### **1.2. Постановка задачи.**

Задача проекта заключается в разработке веб-приложения, которое позволяет пользователям:

- Брать кредиты с предоставлением информации о кредитных условиях;
- Отслеживать статус кредитных заявок и историю взаимодействия с кредиторами;
- Принимать решения по заявкам на кредиты;
- Управлять профилем;
- Анализ кредитной информации на основе статистики;
- Фильтровать и сортировать данные;
- Кроме того, приложение должно обеспечить надежное хранение данных, высокую производительность и устойчивость системы.

### **1.3. Предлагаемое решение.**

Для реализации создается веб-приложение с использованием Vue.js для клиентской части, Flask для серверной части, MongoDB для хранения данных и Docker для контейнеризации и развертывания.

#### **1.4. Качественные требования к решению.**

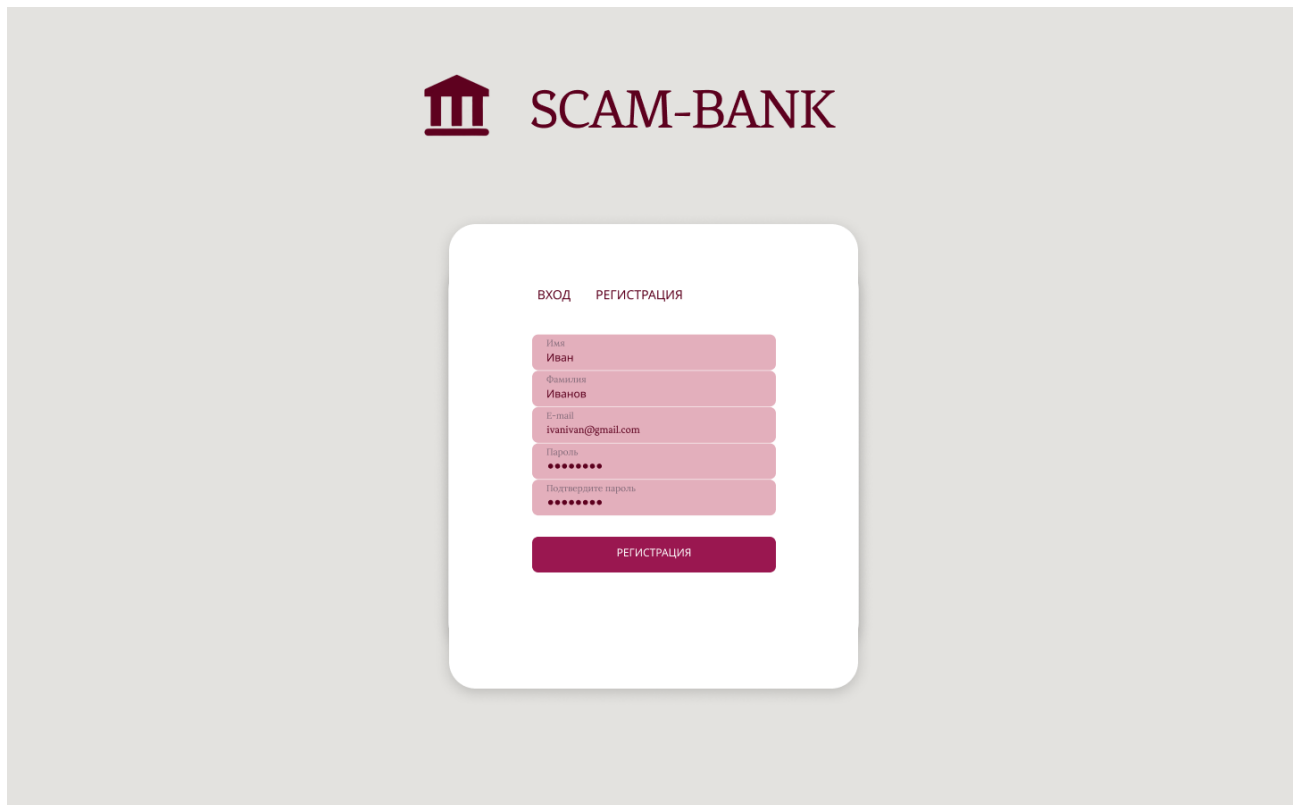
Разрабатываемое веб-приложение должно быть удобным и интуитивно понятным для пользователей, обеспечивать выполнение операций фильтрации, поиска и анализа данных, легко масштабироваться и расширяться за счет модульной архитектуры, а также поддерживать быстрое развертывание на различных платформах благодаря использованию Docker.



## 2. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

### 2.1. Макет UI

Ниже представлен макет приложения (рис. 1-13)



The image shows a UI mockup for a registration screen. At the top, there is a logo consisting of a stylized building icon and the text "SCAM-BANK". Below the logo, there is a white rounded rectangle containing the registration form. The form has two tabs: "ВХОД" (Login) and "РЕГИСТРАЦИЯ" (Registration), with "РЕГИСТРАЦИЯ" being the active tab. The form fields are as follows:

- Имя (Name): Иван
- Фамилия (Surname): Иванов
- E-mail: ivanivan@gmail.com
- Пароль (Password): 8 dots
- Подтвердите пароль (Confirm password): 8 dots

At the bottom of the form, there is a red button labeled "РЕГИСТРАЦИЯ".

Рисунок 1. Регистрация

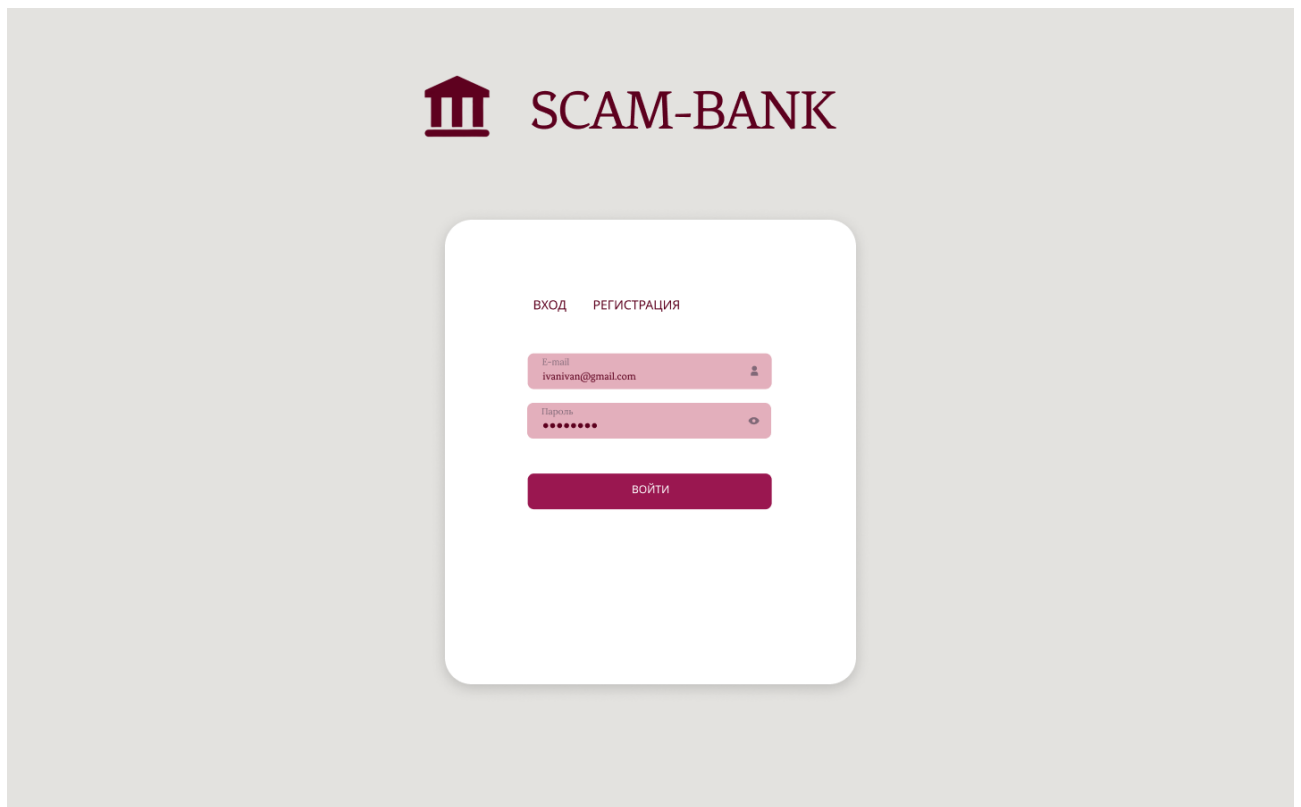


Рисунок 2. Вход

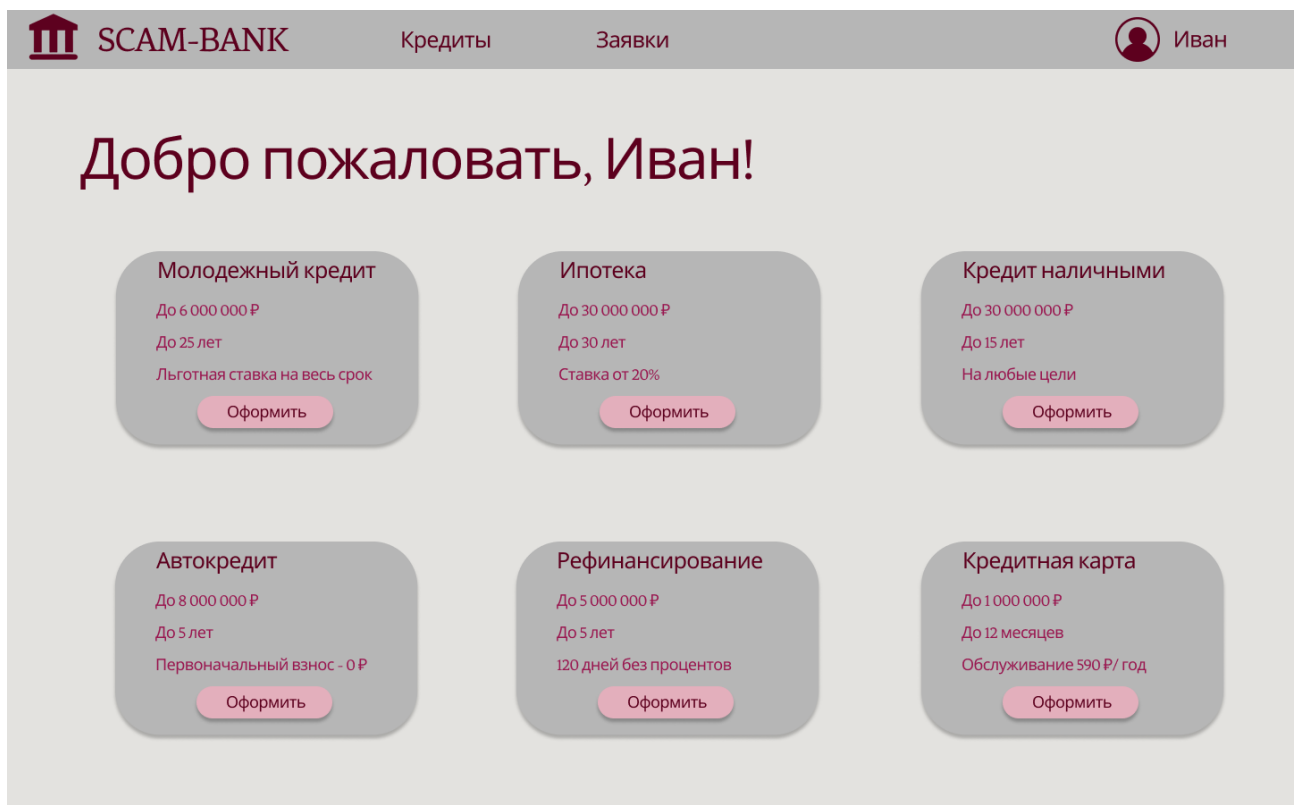


Рисунок 3. Главная страница

Кредиты

Заявки

Выйти

ФИО

E-mail

Место работы

Пол

Пароль

Место работы супруга

Дата рождения

Контактный телефон

Доход

Семейное положение

Количество детей

Доход супруга

Статус самозанятости

Имущество в собственности

Рисунок 4. Профиль

Кредиты

Заявки


Иван

Ваши активные кредиты

Выберите параметр для поиска


Название кредита	Дата открытия	Срок	Сумма	Ставка	Еж. платеж	Дата след. платежа	Задолженность	Просрочено
Ипотека	01.01.2024	10 лет	50 000 000 Р	20%	50 000 Р	15.10.2024	49 000 000 Р	0 платежей

Рисунок 5. Активные кредиты

 SCAM-BANK

Кредиты

Заявки

 Иван

Заполните информацию в профиле или проверьте ее актуальность!

Оформления кредита наличными

Желаемая сумма кредита

100 000 Р

10 000 Р30 000 000 Р

Срок кредита

10 месяцев

3 месяца15 лет

Созаемщики

Контактный телефон

Залог

Сумма залога

Заявка на потребительский кредит

Сумма

100 000 Р

Срок

10 месяцев

Платеж в месяц

от 10 000 Р

Ставка

от 23,9%

Оформить

Рисунок 6. Оформление кредита

 SCAM-BANK

Кредиты

Заявки

 Иван

Ваши заявки

Выберите параметр для поиска

Название кредита	Дата заявки	Статус	Сумма	Ставка	Срок
Ипотека	01.01.2024	В обработке	50 000 000 Р	20%	10 лет

Рисунок 7. Заявки

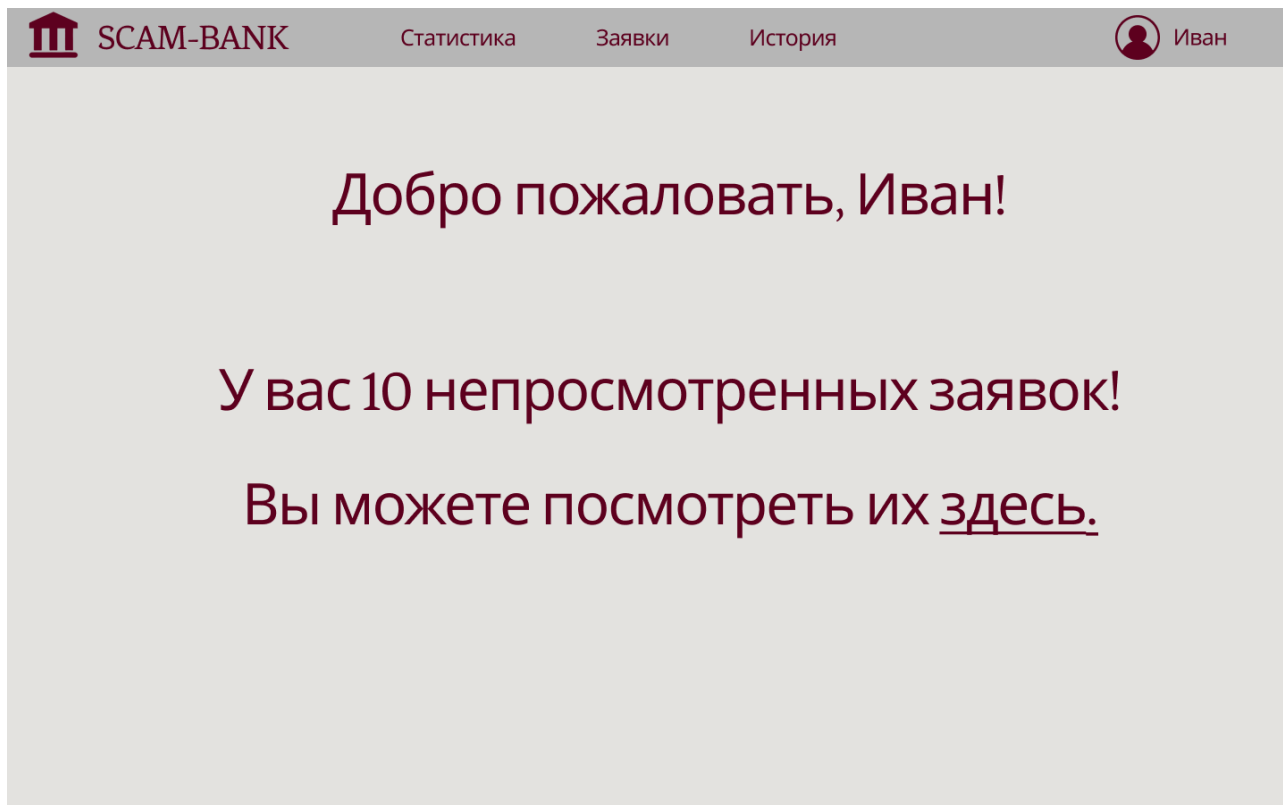


Рисунок 8. Главная страница (сотрудник)

The screenshot displays the user profile page. The header is identical to the previous page, but it includes an additional 'Выйти' (Logout) link with a right-pointing arrow icon on the far right. The main content area features a large circular profile picture placeholder with a person icon and a small edit icon (pencil) at the bottom right. Below the profile picture, there are two columns of form fields. The left column contains: 'ФИО' with a text input and edit icon; 'Пол' with a dropdown menu; 'Дата рождения' with three separate dropdown menus for day, month, and year; and 'Должность' with a text input. The right column contains: 'E-mail' with a text input and edit icon; 'Пароль' with a text input and edit icon; and 'Контактный телефон' with a text input and edit icon.

Рисунок 9. Профиль (сотрудник)


 SCAM-BANK		Статистика	Заявки	История	 Иван			
Кредитная история Иванова Ивана		Контактный телефон: 8 987 654 32 10			E-mail: ivanivan@gmail.com			
Название кредита ▾	Дата открытия ▾	Срок ▾	Сумма ▾	Ставка ▾	Еж. платеж ▾	Задолженность ▾	Просрочено ▾	Статус ▾
Ипотека	01.01.2024	10 лет	50 000 000 ₽	20%	50 000 ₽	49 000 000 ₽	0 платежей	Закрыт
Ипотека	01.01.2024	10 лет	50 000 000 ₽	20%	50 000 ₽	49 000 000 ₽	0 платежей	Открыт

Рисунок 10. Кредитная история (сотрудник)

SCAM-BANK

Статистика

Заявки

История

Иван

Заявки

Q

ФИО

Дата заявки ▾	ФИО ▾	Сумма ▾	Ставка ▾	Срок ▾	Название кредита ▾	Рейтинг ▾	Решение	
01.01.2024	<u>Иванов И.И.</u>	5 000 000 ₽	20%	5 лет	Автокредит	10	<div>✓</div>	<div>✕</div>
01.01.2024	<u>Иванов И.И.</u>	5 000 000 ₽	20%	5 лет	Автокредит	10	<div>✓</div>	<div>✕</div>
01.01.2024	<u>Иванов И.И.</u>	5 000 000 ₽	20%	5 лет	Автокредит	10	<div>✓</div>	<div>✕</div>

Рисунок 11. Заявки (сотрудник)



Рисунок 12. Статистика (сотрудник)

SCAM-BANK

Статистика Заявки История

Иван

### История взаимодействия

Q ФИО

Дата заявки	Дата обработки	ФИО	Сумма	Ставка	Срок	Название кредита	Рейтинг	Решение
01.01.2024	01.01.2024	Иванов И.И.	5 000 000 Р	20%	5 лет	Автокредит	10	Одобрено
01.01.2024	01.01.2024	Иванов И.И.	5 000 000 Р	20%	5 лет	Автокредит	10	Отказано
01.01.2024	01.01.2024	Иванов И.И.	5 000 000 Р	20%	5 лет	Автокредит	10	Отказано

Рисунок 13. История взаимодействия (сотрудник)

## **2.2. Сценарий использования: Регистрация.**

Цель: Зарегистрироваться.

Действующие лица: Клиент, сотрудник банка

Основной сценарий:

1. Пользователь попадает на страницу авторизации (страница 2), затем нажимает кнопку "Регистрация", после чего переходит на страницу регистрации (страница 1).
2. Заполняет следующие поля:
  - Имя
  - Фамилия
  - Email
  - Пароль
  - Подтверждение пароля
3. Нажимает кнопку "Регистрация".

Альтернативный сценарий:

1. Если данная почта уже зарегистрирована - система выводит сообщение об ошибке и просит пользователя ввести данные еще раз.
2. Если пароли не совпадают - система выводит сообщение об ошибке и просит пользователя ввести данные еще раз.

Результат сценария: Пользователь зарегистрирован.

## **2.3. Профиль сотрудника банка.**

Цель:

Получить и заполнить основную информацию о сотруднике банка.

Действующие лица: Сотрудник банка.

Основной сценарий:

1. Сотрудник банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 8).



3. Переходит по нажатию кнопки “Профиль” на панели быстрого доступа к странице профиля сотрудника (страница 9)
4. На странице профиля (страница 9) отображается следующая информация о сотруднике банка: - ФИО - Пол - Возраст - Должность - Контактный телефон - Электронная почта - Пароль
5. Напротив каждого поля (кроме поля “Должность”) есть кнопка “Изменить” для изменения данных профиля.

Альтернативный сценарий:

1. В поля введены некорректные данные - система выводит сообщение о некорректности заполнения поля и просит пользователя ввести данные еще раз.
2. Не все необходимые поля заполнены - система выводит сообщение о том, какие поля необходимо заполнить (являются обязательными к заполнению).

Результат сценария: Получен и заполнен профиль сотрудника с основной информацией о нем.

#### **2.4. Сценарий использования: Просмотр заявок на кредиты.**

Цель: Посмотреть все заявки клиентов банка на кредиты и одобрить/отклонить их.

Действующие лица: Сотрудник банка

Основной сценарий:

1. Сотрудник банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 8).
3. Переходит по гиперссылке на главной странице или по нажатию кнопки “Заявки” на панели быстрого доступа к странице с заявками клиентов (страница 11).
4. На странице заявок клиентов сотрудник может просмотреть все заявки по кредитам (ФИО клиента, дата заявки, сумма кредита, срок, ставка, название кредита, кредитный рейтинг), а также перейти на страницу с его кредитной историей (страница 10)

5. Используя кнопки “Галочка” или “Крестик”, расположенных справа от заявки, сотрудник может принять заявку/отказать в заявке соответственно.
6. Сотрудник вводит в строке поиска ФИО клиента, нажимает кнопку “Поиск”, после чего таблица обновляется с заявками интересующего клиента.
7. Используя кнопку “Сортировка”, можно отсортировать заявки по: - Дате отправки заявки - Кредитному рейтингу - ФИО - Сумме кредита - Сроку кредита - Названию кредита

Альтернативный сценарий:

1. Заявка была обработана другим сотрудником банка в момент просмотра - при попытке взаимодействовать с обработанной заявкой система оповещает сотрудника о том, что заявка уже обработана другим сотрудником.

Результат сценария:

- Обработанная заявка исчезает из списка заявок на странице сотрудника банка (страница 11).
- Клиент получает уведомление об обработанной заявке и результату по ее одобрению на странице заявок (страница 7).
- В кредитной истории клиента появляется запись со статусом “Открыт” в случае одобрения кредита (страница 10).

## **2.5. Сценарий использования: Просмотр кредитной истории клиента.**

Цель: Изучить информацию обо всех открытых и закрытых кредитах конкретного клиента.

Действующие лица: Сотрудник банка

Основной сценарий:

1. Сотрудник банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 8).

3. Переходит по гиперссылке на главной странице или по нажатию значка “Заявки” на панели быстрого доступа к странице с заявками клиентов (страница 11).
4. Переходит нажатием ФИО интересующего клиента по конкретной заявке на кредитную историю (страница 10).
5. На странице кредитной истории в формате таблицы отображены все открытые и закрытые кредиты банка, в том числе следующая информация о них:
  - Название кредита
  - Дата открытия кредита
  - Сумма кредита
  - Срок кредита
  - Ставка кредита
  - Ежемесячный платеж
  - Задолженность
  - Количество просроченных платежей
  - Статус кредита: открыт/закрыт/просрочен

Также на странице (10) отображается сверху контактная информация клиента.

6. Используя кнопку “Сортировка”, можно отсортировать историю по:
  - Названию кредита
  - Дате открытия кредита
  - Сумме кредита
  - Сроку кредита
  - Ставке кредита
  - Ежемесячному платежу
  - Задолженности
  - Количеству просроченных платежей
  - Статусу кредита: открыт/закрыт/просрочен

Альтернативный сценарий:

1. Заявка клиента была одобрена другим сотрудником банка в момент просмотра - при обновлении страницы появляется соответствующая запись в кредитной истории.

Результат сценария: Сотрудник банка изучил кредитную историю клиента.

## **2.6. Сценарий использования: История взаимодействия сотрудника с клиентом.**

Цель: Изучить историю взаимодействия сотрудника с клиентами банка.

Действующие лица: Сотрудник банка

Основной сценарий:

1. Сотрудник банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 8).
3. Переходит на страницу (страница 13) истории взаимодействия по нажатию значка “История” на панели быстрого доступа.
4. На странице отображены все обработанные сотрудником заявки клиентов.

Информация об истории хранится в таблице со следующими полями:

- Дата заявки
  - Дата обработки
  - ФИО
  - Сумма кредита
  - Ставка кредита
  - Срок кредита
  - Название кредита
  - Кредитный рейтинг
  - Решение по кредиту (одобрено/отказано)
5. Используя кнопку “Сортировка”, можно отсортировать историю по:
    - Дате заявки
    - Дате обработки
    - ФИО

- Сумме кредита
- Ставке кредита
- Сроку кредита
- Названию кредита
- Кредитному рейтинг
- Решению по кредиту (одобрено/отказано)

Результат сценария: Сотрудник имеет доступ к истории обработок заявок.

## **2.7. Сценарий использования: Статистика банка**

Цель: Получение статистики банка по пользователям и типам кредита.

Действующие лица: Сотрудник банка.

Основной сценарий:

1. Сотрудник банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 8).
3. Переходит по нажатию кнопки “Статистика” на панели быстрого доступа к странице статистики банка (страница 12).
4. На странице статистики (страница 12) сотрудник выбирает параметр, по которому будет строиться статистика. Выбор параметров осуществляется по трем ключам:
  - Пользователь/сотрудник: работа со статистикой пользователей банка или его сотрудников.
  - Пользователь: есть возможность выбрать всех пользователей банка или одного конкретного.
  - Тип кредита: есть возможность выбрать все типы кредитов банка или один конкретный.
5. Нажав на кнопку "Фильтр", имеется возможность выбрать следующие данные для построения графиков:
  - Дата открытия кредита
  - Сумма кредита

- Срок кредита
  - Ежемесячный платеж
  - Статус кредита (Открыт/закрыт/просрочен)
6. Используя кнопку “Построить”, строится в области для графика статистика по выбранным параметрам и данным.

Альтернативный сценарий:

1. Данные по выбранным параметрам статистики отсутствуют - выводится сообщение об отсутствии данных в БД.

Результат сценария: Выводится статистика по кредитам банка по выбранным параметрам и данным для построения.

## **2.8. Сценарий использования: Массовый импорт/экспорт данных**

Цель: Загрузка и выгрузка данных банка по клиентам и их кредитам.

Действующие лица: Сотрудник банка

Основной сценарий:

1. Сотрудник банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 8).
3. Переходит по нажатию кнопки “Статистика” на панели быстрого доступа к странице статистики банка (страница 12).
4. На странице статистики (страница 12) имеется две кнопки для импорта/экспорта данных: “Импортировать” и “Экспортировать”.
5. При нажатии на кнопку “Импортировать”:
  - Появляется окно с выбором файла для загрузки в БД в поддерживаемом формате (.csv, .json).
  - Выбранный файл загружается в БД.
6. При нажатии на кнопку “Экспортировать”:
  - Выгружаются данные с БД со сведениями об клиентах и их кредитах.

Альтернативный сценарий:

1. Выбран неподдерживаемый формат файла для импорта - выводится сообщение об ошибке и информация об поддерживаемых форматах файла.

Результат сценария: Загружены/выгружены данные банка по клиентам и их кредитам.

## **2.9. Сценарий использования: Профиль клиента**

Цель: Получить и заполнить основную информацию о клиенте банка.

Действующие лица: Клиент банка.

Основной сценарий:

1. Клиент банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 3).
3. Переходит по нажатию кнопки “Профиль” на панели быстрого доступа к странице профиля клиента (страница 4)
4. На странице профиля (страница 4) отображается следующая информация о клиенте:
  - ФИО
  - Пол
  - Возраст
  - Доход
  - Статус самозанятости
  - Место работы
  - Семейное положение
  - Место работы супруга
  - Доход супруга (если есть)
  - Наличие детей
  - Имущество в собственности
  - Контактный телефон

- Электронная почта
  - Пароль
5. Напротив каждого поля есть кнопка “Изменить” для изменения данных профиля.

Альтернативный сценарий:

1. В поля введены некорректные данные - система выводит сообщение о некорректности заполнения поля и просит пользователя ввести данные еще раз.
2. Не все необходимые поля заполнены - система выводит сообщение о том, какие поля необходимо заполнить (являются обязательными к заполнению).

Результат сценария: Получен и заполнен профиль клиента банка.

## **2.10. Сценарий использования: Просмотр открытых кредитов**

Цель: Изучить информацию обо всех открытых кредитах клиента.

Действующие лица: Клиент банка.

Основной сценарий:

1. Клиент банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 3).
3. Переходит по нажатию кнопки “Кредиты” на панели быстрого доступа к странице с информацией обо всех открытых кредитах пользователя (страница 5).
4. На странице с кредитами (страница 5) отображается список открытых кредитов клиента, о каждом из которых присутствует следующая информация:
  - Название кредита
  - Дата открытия
  - Срок
  - Сумма
  - Ставка
  - Ежемесячный платеж
  - Дата следующего платежа
  - Суммарная задолженность



- Количество просроченных платежей
- 5. Используя кнопку “Сортировка”, можно отсортировать кредиты по:
  - Названию кредита
  - Дате открытия
  - Сроку
  - Сумме
  - Ставке
  - Ежемесячному платежу
  - Дате следующего платежа
  - Суммарной задолженности
  - Количество просроченных платежей
- 6. Используя объект combobox “Поиск”, можно найти кредит по заданному параметру. Если выбран параметр “Дата открытия”, то для ввода даты будет доступно текстовое поле, куда надо будет указать дату в формате ДД.ММ.ГГГГ, при выборе остальных параметров для ввода будет доступно обычное текстовое поле.

Альтернативный сценарий:

Пользователь ввел некорректные данные в поле “Поиск” или не выбрал параметр (например, ввел буквы в поле, где надо указать сумму кредита, ежемесячный платеж и т.д.) - система выдает сообщение об ошибке и просит пользователя ввести данные еще раз.

Результат сценария: Клиент банка ознакомился со своей кредитной историей.

## **2.11. Сценарий использования: Оформление кредита**

Цель: Оформить кредит для клиента банка.

Действующие лица: Клиент банка

Основной сценарий:

1. Клиент банка авторизуется (страница 2) или регистрируется (страница 1).

2. Переходит на главную страницу (страница 3).
3. На главной странице клиент находит среди доступных кредитов интересующий вариант, затем нажимает на этот вариант и переходит к странице оформления кредита (страница 6).
4. На странице оформления кредита клиент заполняет следующие поля:
  - Сумма кредита
  - Срок кредита
  - Созаемщик (опционально)
  - Залог (опционально)
5. После выбора суммы и срока кредита клиент нажимает на кнопку “Оформить”, после чего отправляется заявка на кредит.

Альтернативный сценарий:

1. В случае, если профиль клиента заполнен не полностью (отсутствует контактный телефон, почта и т.д.) - система выдает сообщение об этом и просит пользователя ввести данные в профиль.
2. В случае, если клиент указал данные в поле “Созаемщик” или “Залог” - пользователю предоставляются улучшенные условия кредита (уменьшение процента ставки).

Результат сценария: Клиент банка отправил заявку на оформление кредита.

## **2.12. Сценарий использования: Заявки на оформление кредита клиента.**

Цель:

Просмотр заявок на оформление кредита клиента.

Действующие лица: Клиент банка.

Основной сценарий:

1. Клиент банка авторизуется (страница 2) или регистрируется (страница 1).
2. Переходит на главную страницу (страница 3).

3. Переходит по нажатию кнопки “Заявки” на панели быстрого доступа к странице с информацией обо всех заявках на оформление кредитов пользователя (страница 7).

4. На странице заявок представлен список активных заявок клиента, о каждой из которых присутствует следующая информация:

- Дата отправки заявки
- Статус заявки (в обработке / одобрено / отклонено)
- Название кредита
- Срок кредита
- Сумма кредита
- Ставка кредита

5. Используя кнопку “Сортировка”, можно отсортировать заявки по:

- Дате отправки заявки
- Статусу заявки (в обработке / одобрено / отклонено)
- Названию кредита
- Сроку кредита
- Сумме кредита
- Ставке кредита

6. Используя объект combobox “Поиск”, можно найти заявку по заданному параметру. Если выбран параметр “Дата открытия”, то для ввода даты будет доступно текстовое поле, куда надо будет указать дату в формате ДД.ММ.ГГГГ, при выборе остальных параметров для ввода будет доступно обычное текстовое поле.

Альтернативный сценарий: Пользователь ввел некорректные данные в поле “Поиск” или не выбрал параметр (например, ввел буквы в поле, где надо указать сумму кредита, ежемесячный платеж и т.д.) - система выдает сообщение об ошибке и просит пользователя ввести данные еще раз.

Результат сценария: Клиент банка проверил заявки на оформление кредита.

## 2.13. Вывод

На основании описанных сценариев использования можно сделать вывод, что операции чтения будут преобладать над операциями записи в данном приложении. Пользователи смогут просматривать свои заявки, свои открытые кредиты, свой профиль. Администраторы могут просматривать заявки, кредитную историю, анализировать статистику, просматривать историю взаимодействия. Пользователю доступны следующие операции записи: регистрация, оформление заявки на кредит, заполнение профиля. Администратор же имеет возможность одобрять/отклонять заявки. Таким образом, основная функция приложения — обеспечение быстрого и эффективного доступа к данным.

### 3. МОДЕЛЬ ДАННЫХ.

#### 3.1. Нереляционная модель данных.

Графическое представление нереляционной модели базы данных MongoDB представлена на рисунке 14:

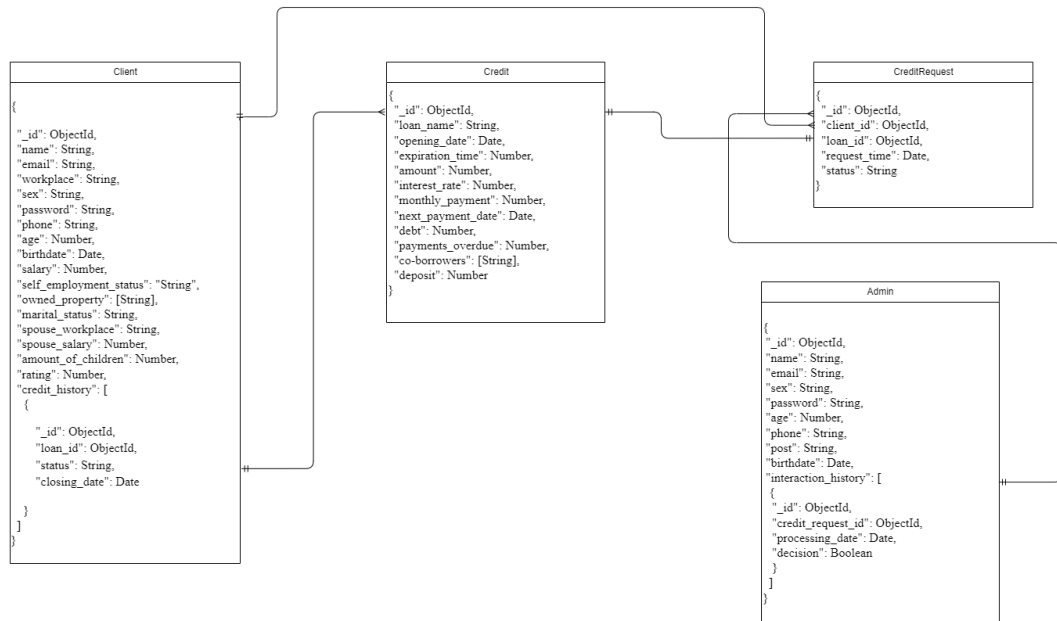


Рисунок 14 - Графическое представление нереляционной модели.

#### Коллекции и сущности.

n - максимально необходимый размер строки для текста. Оптимальное n = 150 СИМВОЛОВ.

Client. Хранение информации о клиенте.

```
{
  "_id": ObjectId, // 12
  "name": String, // n
  "email": String, // n
  "workplace": String, // n
  "sex": String, // n
  "password": String, // n
  "phone": String, // n
  "age": Number, // 8
  "birthdate": Date, // 8
  "salary": Number, // n
  "self_employment_status": String, // n "self-employed",
```

```

"entrepreneur", "idle"
  "owned_property": [String], // 10n
  "marital_status": String, // n "married", "single"
  "spouse_workplace": String, // n
  "spouse_salary": Number, // 8
  "amount_of_children": Number, // 8
  "rating": Number, // 8
  "credit_history": [
  {
    "_id": ObjectId, // 12
    "loan_id": ObjectId, // 12
    "status": String, // n "opened", "expired", "closed"
    "closing_date": Date // 8
  }
  ]
}

```

**Admin.** Хранение информации о сотруднике банка (администратора).

```

{
  "_id": ObjectId, // 12
  "name": String, // n
  "email": String, // n
  "sex": String, // n
  "password": String, // n
  "age": Number, // 8
  "phone": String, // n
  "post": String, // n
  "birthdate": Date, // 8
  "interaction_history": [
    {
      "_id": ObjectId, // 12
      "credit_request_id": ObjectId, // 12
      "processing_date": Date, // 8
      "decision": Boolean // 1
    }
  ]
}

```

**Credit.** Хранение данных о кредите.

```

{
  "_id": ObjectId, // 12
  "loan_name": String, // n
  "opening_date": Date, // 8
  "expiration_time": Number, // 8
  "amount": Number, // 8
  "interest_rate": Number, // 8
  "monthly_payment": Number, // 8
  "next_payment_date": Date, // 8
  "debt": Number, // 8
}

```

```

"payments_overdue": Number, // 8
"co-borrowers": [String], // 5n
"deposit": Number // 8
}

```

**CreditRequest.** Хранение информации о кредитных запросах пользователей.

```

{
  "_id": ObjectId, // 12
  "client_id": ObjectId, // 12
  "loan_id": ObjectId, // 12
  "request_time": Date, // 8
  "status": String // n "processing", "approved", "rejected"
}

```

### Оценка объема информации.

Средний размер информации, хранимой в модели:

- Client:  $12 * 3 + 8 * 6 + n * 20 = 3084$  байт
- Credit:  $8 * 9 + 12 + n * 6 = 984$  байт
- CreditRequest:  $8 + 12 * 3 + n = 194$  байт
- Admin:  $1 + 8 * 3 + 12 * 3 + n * 6 = 961$  байт

В среднем пользователь будет:

- брать 5 кредитов.
- отправлять количество заявок, равному 8.
- каждая заявка будет обработана и хранится в истории взаимодействия с клиентом: 5.

При количестве клиентов, равных clients:

$$V(\text{clients}) = (3084 + 5 \cdot (984 + 961) + 8 \cdot 194) \cdot \text{clients} = 14361 \cdot \text{clients}$$

### Избыточность данных

В таблице Client избыточными является поля self\_employment\_status, age, marital\_status - 168 байт. В таблице Admin избыточным является поле age - 8 байт. В таблице Credit избыточным является поле monthly\_payment, next\_payment\_date - 16 байт.

Тогда формула для чистого объема данных примет вид (см. рисунок 15):

$$V_{\text{clear}}(\text{clients}) = (2916 + 5 \cdot (968 + 953) + 8 \cdot 194) \cdot \text{clients} = 14073 \cdot \text{clients}.$$

Рисунок 15 - формула чистого объема данных.

Тогда избыточность данных (отношение между фактическим объемом модели и "чистым" объемом данных) (см. рисунок 16):

$$\frac{V(\text{clients})}{V_{\text{clear}}(\text{clients})} = \frac{14361 \cdot \text{clients}}{14073 \cdot \text{clients}} = 1.0204$$

Рисунок 16 — Избыточность данных.

## Запросы к модели, с помощью которых реализуются сценарии использования.

Авторизация пользователя.

```
user = db.Client.find_one({
    "email": "user@example.com",
    "password": "user_password"
})
```

Регистрация пользователя.

```
db.Client.insert_one({
    "name": "Иван Иванов",
    "email": "ivanov@example.com",
    "workplace": "Программист",
    "sex": "male",
    "password": "secret123",
    "phone": "1234567890",
    "age": 30,
    "birthdate": "1993-05-15T00:00:00Z",
    "salary": 60000,
    "self_employment_status": True,
    "owned_property": ["Квартира"],
    "marital_status": "single",
    "spouse_workplace": "Офис менеджер",
    "spouse_salary": 50000,
    "amount_of_children": 2,
    "rating": 7.75,
    "credit_history": []
})
```

Авторизация администратора.



```
admin = db.Admin.find_one({
    "email": "admin@example.com",
    "password": "admin_password"
})
```

### Регистрация администратора.

```
db.Admin.insert_one({
    "name": "Кузнецов А.Н.",
    "email": "kuznetsov@example.com",
    "sex": "male ",
    "password": "password123",
    "age": 32,
    "phone": "3216549870",
    "post": "Администратор",
    "birthdate": "1991-04-12T00:00:00Z",
    "interaction_history": []
})
```

### Поиск заявки по кредиту.

```
credit_request = db.CreditRequest.find_one({
    "client_id": ObjectId("client_id"),
    "loan_id": ObjectId("loan_id")
})
```

### Создание заявки по кредиту.

```
db.CreditRequest.insert_one({
    "client_id": ObjectId("client_id"),
    "loan_id": ObjectId("loan_id"),
    "request_time": "2024-10-10T12:00:00Z",
    "status": "processing"
})
```

### Создание нового кредита.

```
db.Credit.insert_one({
    "loan_name": "Ипотека",
    "opening_date": "2024-01-15T09:00:00Z",
    "expiration_time": 36,
    "amount": 500000.00,
    "interest_rate": 5.00,
    "monthly_payment": 14999.99,
    "next_payment_date": "2024-11-15T00:00:00Z",
    "debt": 300000.00,
    "payments_overdue": 1,
    "co-borrowers": None,
})
```

```
"deposit": None
})
```

**Вынесение решения по кредиту: одобрение.**

```
db.CreditRequest.update_one(
{ "_id": ObjectId("request_id") },
{ "$set": { "status": "approved" } }
)
```

**Вынесение решения по кредиту: отклонение.**

```
db.CreditRequest.update_one(
{ "_id": ObjectId("request_id") },
{ "$set": { "status": "rejected" } }
)
```

**Изменение данных в личном кабинете пользователя: изменение номера телефона и зарплаты.**

```
db.Client.update_one(
{ "_id": ObjectId("user_id") },
{ "$set": { "phone": "9876543210", "salary": 65000 } }
)
```

**Изменение данных в личном кабинете пользователя: вычисление и запись возраста.**

```
client_collection = db["Client"]

current_date = datetime.now()

for client in client_collection.find({"birthdate": {"$exists": True}}):
    birthdate = client["birthdate"]
    age = current_date.year - birthdate.year - ((current_date.month,
current_date.day) < (birthdate.month, birthdate.day))

    client_collection.update_one(
        {"_id": client["_id"]},
        {"$set": {"age": age}}
    )
```

**Изменение данных в личном кабинете администратора: изменение номера телефона и поста.**

```
db.Admin.update_one(
{ "_id": ObjectId("admin_id") },
{ "$set": { "phone": "9876543210", "post": "Старший администратор" } }
)
```

**Изменение данных в личном кабинете администратора: вычисление и запись возраста.**

```
admin_collection = db["Admin"]

for admin in admin_collection.find({"birthdate": {"$exists": True}}):
    birthdate = admin["birthdate"]
    age = current_date.year - birthdate.year - ((current_date.month,
current_date.day) < (birthdate.month, birthdate.day))

    admin_collection.update_one(
        {"_id": admin["_id"]},
        {"$set": {"age": age}}
    )
```

**Изменение статуса кредита в кредитной истории.**

```
db.Client.update_one(
{ "_id": ObjectId("client_id"), "credit_history.loan_id":
ObjectId("loan_id") },
{ "$set": { "credit_history.$.status": "closed" } }
)
```

**Просмотр статистики по кредитам пользователя: кредиты с долгом больше 100,000**

```
pipeline = [
    { "$match": { "_id": ObjectId("client_id") } },
    { "$unwind": "$credit_history" },
    {
        "$lookup": {
            "from": "Credit",
            "localField": "credit_history.loan_id",
            "foreignField": "_id",
            "as": "credit_info"
        }
    },
    { "$unwind": "$credit_info" },
    { "$match": { "credit_info.debt": { "$gt": 100000 } } },
    { "$project": { "name": 1, "credit_info.loan_name": 1,
"credit_info.debt": 1 } }
]
```

```
stats = list(db.Client.aggregate(pipeline))
```

## Массовый импорт из бд

```
mongoimport --db your_db --collection Client --file path_to_file.json -  
-jsonArray
```

## Массовый экспорт в бд

```
mongoexport --db your_db --collection Client --out path_to_output.json  
--jsonArray
```

## 3.2. Реляционная модель.

Графическое представление реляционной модели базы данных PostgreSQL представлена на рисунке 17:

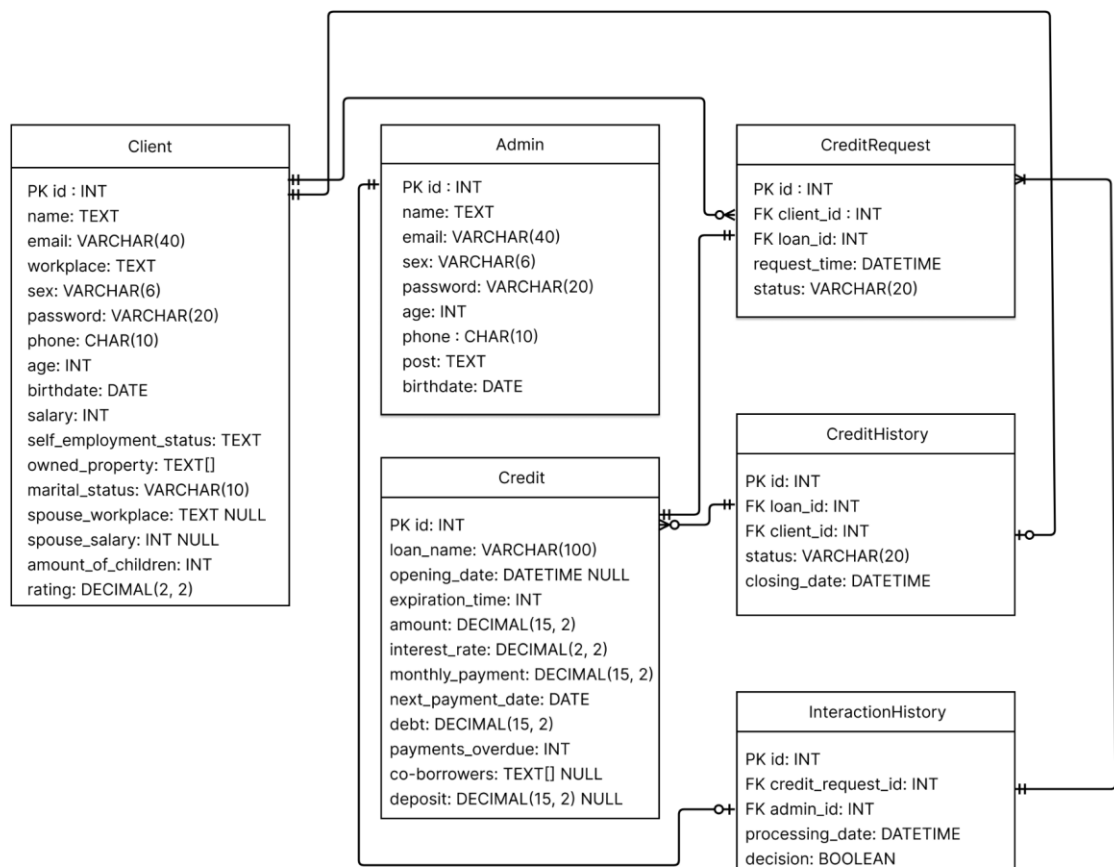


Рисунок 17 - Реляционная модель данных (PostgreSQL).

## Коллекции и сущности.

n - максимально необходимый размер строки для текста. Оптимальное n = 150 СИМВОЛОВ.

Client. Хранение информации о клиенте.

```
id int(4)
name text(n)
email varchar(40)
workplace text(n)
sex varchar(6)
password varchar(20)
phone char(10)
age int(4)
birthdate date(8)
salary int(4)
self_employment_status varchar(13): "self-employed", "entrepreneur",
"idle"
owned_property text[10] (10*n)
marital_status varchar(10): "married", "single"
spouse_workplace text(n)
spouse_salary int(4)
amount_of_children int(4)
rating decimal(4)
```

Credit. Хранение данных о кредите.

```
id int(4)
loan_name varchar(100)
opening_date datetime(8)
expiration_time int(4)
amount decimal(17)
interest_rate decimal(4)
monthly_payment decimal(17)
next_payment_date date(8)
debt decimal(17)
payments_overdue int(4)
co-borrowers: text[5] (5*n)
deposit: decimal(17)
```

CreditHistory. Хранение данных о кредитной истории конкретного клиента.

```
id int(4)
client_id int(4)
loan_id int(4)
status varchar(20): "opened", "closed", "expired"
closing_date datetime(8)
```

**CreditRequest.** Хранение данных о заявке на кредит.

```
id int(4)
client_id int(4)
loan_id int(4)
request_time datetime(8)
status varchar(20): "processing", "approved", "rejected"
```

**Admin.** Хранение данных о сотруднике банка.

```
id int(4)
name text(n)
email varchar(40)
sex varchar(6)
password varchar(20)
age int(4)
phone char(10)
post text(n)
birthdate date(8)
```

**InteractionHistory.** Хранение данных о взаимодействии администратора с заявками по кредитам.

```
id int(4)
credit_request_id int(4)
admin_id int(4)
processing_date datetime(8)
decision boolean(1)
```

### **Оценка объема информации.**

Средний размер информации, хранимой в модели:

- Client:  $13 + 4 * 6 + 6 + 8 + 10 * 2 + 20 + 40 + n * 13 = 2081$  байт
- Credit:  $4 * 4 + 8 * 2 + 5 * n + 17 * 4 + 100 = 950$  байт
- CreditRequest:  $4 * 3 + 8 + 20 = 40$  байт
- CreditHistory:  $4 * 3 + 20 = 32$  байт
- InteractionHistory:  $1 + 4 * 3 + 8 = 21$  байт
- Admin:  $4 * 2 + 6 + 8 + 10 + 20 + 40 + n * 2 = 392$  байт

В среднем пользователь будет:

- брать 5 кредитов.

- отправлять количество заявок, равному 8.
- хранить в истории как минимум столько кредитов, сколько он взял: 5
- каждая заявка будет обработана и хранится в истории взаимодействия с клиентом: 5.
- в среднем заявки обрабатывает 4 сотрудника банка.

При количестве клиентов, равных clients:

$$V(\text{clients}) = (2081 + 5 \cdot (950 + 32 + 21) + 8 \cdot 40 + 392 \cdot 4) \cdot \text{clients} = 8984 \cdot \text{clients}$$

### **Избыточность данных.**

В таблице Client избыточными являются поля self\_employment\_status, age, marital\_status - 27 байт В таблице Admin избыточным является поле age - 4 байта В таблице Credit избыточным является поле monthly\_payment, next\_payment\_date - 25 байт.

Тогда формула для чистого объема данных примет вид:

$$V_{\text{clear}}(\text{clients}) = (2053 + 5 \cdot (925 + 32 + 21) + 8 \cdot 40 + 388 \cdot 4) \cdot \text{clients} = 8897 \cdot \text{clients}.$$

Тогда избыточность данных (отношение между фактическим объемом модели и "чистым" объемом данных):

$$V(\text{clients}) / V_{\text{clear}}(\text{clients}) = 8984 \cdot \text{clients} / 8897 \cdot \text{clients} = 1.0097.$$

### **Запросы к модели, с помощью которых реализуются сценарии использования.**

#### **Авторизация пользователя**

```
SELECT COUNT(*)
FROM Client
WHERE id = $id
AND password = $password;
```

#### **Регистрация пользователя**

```
SELECT COUNT(*)
FROM Client
WHERE email = $email;
```

### Авторизация администратора

```
SELECT COUNT(*)
FROM Admin
WHERE id = $id
AND password = $password;
```

### Регистрация администратора

```
SELECT COUNT(*)
FROM Admin
WHERE email = $email;
```

### Поиск заявки по кредиту

```
SELECT *
FROM CreditRequest
WHERE status = $status;
```

### Создание заявки по кредиту

```
INSERT INTO CreditRequest (FK_client_id, FK_loan_id, request_time,
status)
VALUES (1, 1, '2024-10-10 12:00:00', 'processing');
```

### Создание нового кредита

```
INSERT INTO Credit (
    loan_name,
    opening_date,
    expiration_time,
    amount,
```



```
        interest_rate,  
        monthly_payment,  
        next_payment_date,  
        debt,  
        payments_overdue,  
        co-borrowers,  
        deposit  
    ) VALUES (  
        'Ипотека',  
        '2024-10-10 12:00:00',  
        24,  
        400000.00,  
        16.00,  
        17500.00,  
        '2024-11-10',  
        200000.00,  
        1122334455,  
        ['Иванов И.И.'],  
        80000.00  
    );
```

### Вынесение решения по кредиту: одобрение

```
UPDATE CreditRequest  
SET status = 'approved'  
WHERE id = 1;
```

### Вынесение решения по кредиту: отклонение

```
UPDATE CreditRequest  
SET status = 'rejected'  
WHERE id = 1;
```

### Изменение данных в личном кабинете пользователя: изменение номера телефона.

```
UPDATE Client  
SET phone = '9876543210'  
WHERE id = 1;
```

Изменение данных в личном кабинете пользователя: вычисление и запись возраста.

```
UPDATE Client
  SET age = EXTRACT(YEAR FROM AGE(birthdate));
```

Изменение данных в личном кабинете администратора: изменение поста.

```
UPDATE Admin
  SET post = 'Старший администратор'
  WHERE id = 1;
```

Изменение данных в личном кабинете администратора: вычисление и запись возраста.

```
UPDATE Admin
  SET age = EXTRACT(YEAR FROM AGE(birthdate));
```

Изменение статуса кредита в кредитной истории.

```
UPDATE CreditHistory
  SET status = 'closed'
  WHERE id = 1;
```

Просмотр информации о кредитной истории

```
SELECT
  ch.id AS credit_history_id,
  c.name AS client_name,
  c.email AS client_email,
  cr.loan_name AS loan_name,
  cr.amount AS loan_amount,
  cr.interest_rate AS loan_interest_rate,
  cr.monthly_payment AS monthly_payment,
  ch.status AS credit_status,
  cr.expiration_time AS loan_duration
FROM
```

```
CreditHistory ch
JOIN
  Client c ON ch.FK_client_id = c.id
JOIN
  Credit cr ON ch.FK_loan_id = cr.id
WHERE
  cr.expiration_time < 12;
```

## Массовый импорт из бд

```
COPY Client(id, name, email, workplace, sex, password, phone, age,
birthdate, salary, self_employment_status, owned_property,
marital_status, spouse_workplace, spouse_salary, amount_of_children,
rating)
FROM '/path/to/your/client_data.csv'
DELIMITER ','
CSV HEADER;
```

## Массовый экспорт в бд

```
COPY Client(id, name, email, workplace, sex, password, phone, age,
birthdate, salary, self_employment_status, owned_property,
marital_status, spouse_workplace, spouse_salary, amount_of_children,
rating)
TO '/path/to/exported/client_data.csv'
DELIMITER ','
CSV HEADER;
```

## 3.3. Сравнение моделей

### Удельный объем информации

Нереляционная модель имеет больший удельный объем информации по сравнению с реляционной. Это связано с тем, что в нереляционной модели данные часто дублируются для повышения скорости доступа, в то время как реляционная модель использует нормализацию для уменьшения избыточности.

### Сравнения для моделей

- Для модели Plants: 4048 байт (NoSQL) vs 4048 байт (SQL)
- Для модели User: 44068 байт (NoSQL) vs 2572 байт (SQL)

- Для модели Trade: 2080 байт (NoSQL) vs 56 байт (SQL)
- Для модели CareRules: 1032 байт (NoSQL) vs 1038 байт (SQL)

### **Запросы по отдельным юзкейсам.**

Количество запросов для выполнения операций в нереляционной модели будет меньше, чем в реляционной, особенно при работе с большими объемами данных.

### **Сравнения количества запросов для отдельных юзкейсов**

- Просмотр покупок/обменов: 1 запрос (NoSQL) vs 2 запроса (SQL)
- Создание запроса на покупку/обмен: 1 запрос (NoSQL) vs 1 запрос (SQL)
- Авторизация/Регистрация: 1 запрос (NoSQL) vs 1 запрос (SQL)
- Поиск растения: 1 запрос (NoSQL) vs 1 запрос (SQL)
- Количество задействованных коллекций
- Просмотр покупок/обменов: 1 (NoSQL) vs 4 (SQL)
- Создание запроса на покупку/обмен: 1 (NoSQL) vs 1 (SQL)
- Авторизация/Регистрация: 1 (NoSQL) vs 1 (SQL)
- Поиск растения: 1 (NoSQL) vs 1 (SQL)

### **Вывод**

На основе проведенного анализа можно сделать следующие выводы:

- Удельный объем информации: Реляционная модель демонстрирует меньший объем данных по сравнению с нереляционной моделью.
- Количество запросов: Нереляционная модель показывает лучшие результаты по количеству запросов для большинства операций.
- Число используемых коллекций: Нереляционная модель требует меньше коллекций для выполнения аналогичных операций.
- Упрощение масштабирования: Нереляционные модели часто проще масштабировать горизонтально, что делает их более подходящими для распределенных систем.

Так как процесс оформления кредита - процесс не самый быстрый (необходимо анализировать профиль, оценивать рейтинг клиента и его кредитную историю),

то можно пожертвовать быстродействием системы в пользу меньшего объема памяти (что особенно актуально для больших банков). Кроме того, при масштабировании системы понадобится придумывать более гибкие, компактные коллекции нереляционной модели. Таким образом, предпочтительнее выглядит использование реляционной модели.

## **4. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ**

### **4.1. Краткое описание приложения.**

Frontend реализован с использованием фреймворка Vue.js. Он осуществляет взаимодействие с backend через GET и POST запросы, получая и отправляя данные в формате JSON. На frontend отображаются полученные данные в удобной для пользователя в форме таблиц и других визуальных элементов.

Backend реализован на языке Python с использованием фреймворка Flask. Взаимодействие с frontend происходит через API, реализующее обработку GET и POST запросов. С помощью GET запросов frontend может получать информацию о кредитах или пользователях, а POST запросы используются для отправки данных, таких как заявки на кредиты или обновления данных клиента.

Для упрощения развертывания и обеспечения изоляции окружения все компоненты приложения упакованы в контейнеры с использованием Docker. Это решение позволяет легко развертывать приложение в различных средах и гарантировать совместимость между всеми компонентами системы.

### **4.2. Используемые технологии.**

Frontend: Vue.

Backend: Flask (Python).

DB: MongoDB.

### **4.3. Схема экранов приложения.**

На рисунке 18 представлена схема экранов приложения.

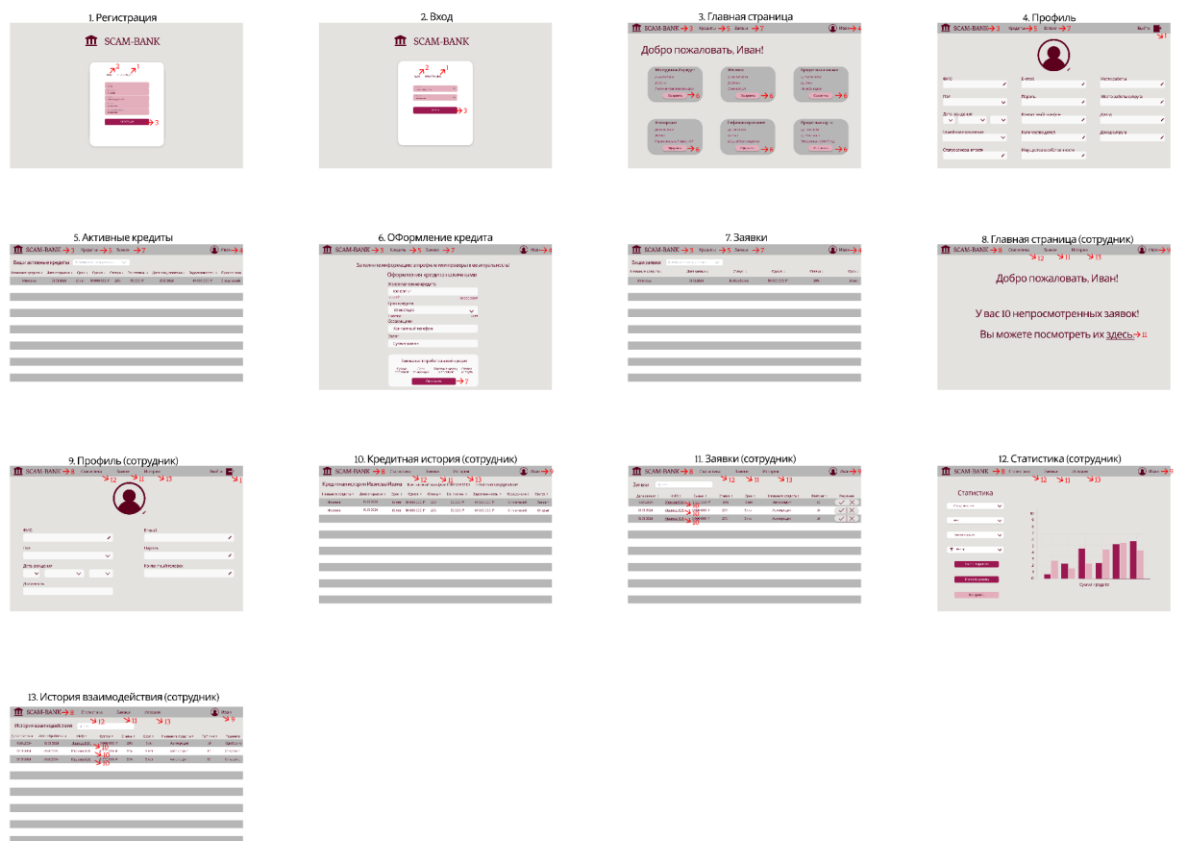


Рисунок 18. Схема экранов приложения.

## **5. ВЫВОДЫ**

### **5.1. Достигнутые результаты.**

По итогу выполнения работы реализовано веб-приложение для выдачи, отслеживания и получения кредитов “Scam Bank”.

Клиенты банка могут оформлять заявки на кредит, просматривать свои заявки и их статус, отслеживать открытые кредиты, сортировать и фильтровать информацию по каждому полю. Кроме того, реализован профиль клиента с возможностью редактирования. Сотрудники банка (администраторы) имеют страницу для просмотра кредита, могут отслеживать все поданные заявки, свою историю взаимодействия с пользователями, а также просматривать, изучать, сортировать, фильтровать кредитную историю каждого конкретного пользователя. Кроме того, реализован профиль администратора с возможностью редактирования.

Разработанное приложение поддерживает импорт/экспорт базы данных в формате json, обеспечивая возможность дополнительного контроля над базой данных.

### **5.2. Недостатки и пути для улучшения полученного решения.**

На данный момент приложение “Scam Bank” имеет недостатки, устранение которых позволит улучшить пользовательский опыт и функциональность приложения:

1. Малое количество вариантов кредитных тарифов.

Приложение имеет только 6 доступных тарифов, данного количества явно недостаточно. Для решения данной проблемы нужно добавить большее количество кредитных тарифов.

2. Ограниченная языковая поддержка



Приложение доступно только на русском языке, что сужает его потенциальную аудиторию. Реализация многоязычности расширит возможности приложения и сделает его доступным для международных пользователей.

### 3. Проблема отсутствия автоматизации одобрения кредитов.

В текущей версии администратор должен вручную одобрять или отклонять заявки, что будет затруднительно при большом числе пользователей. Для решения этой проблемы можно реализовать систему автоматической оценки заявки на кредит.

### 4. Отсутствие мобильного приложения.

В данный момент приложение доступно только в веб-формате, что ограничивает удобство использования на мобильных устройствах. Разработка нативных мобильных приложений для платформ iOS и Android позволит пользователям комфортно работать с приложением на мобильных устройствах.

### 5. Отсутствие технической поддержки

В данный момент отсутствует возможность обращения пользователя в техническую поддержку. Для решения данной проблемы нужно добавить интерфейс для консультаций и обратной связи с клиентами

### 6. Отсутствие дополнительных систем безопасности

В данный момент в приложении отсутствуют любые дополнительные системы безопасности, такие как двухфакторная аутентификация. Ее добавление обеспечит гарантию безопасности для пользователей.

## 5.3. Будущее развитие решения.

В рамках дальнейшего развития веб-приложения "Scam Bank" планируется внедрение следующего функционала:

1. Расширение кредитных тарифов. Приложение будет оснащено большим выбором кредитных тарифов, чтобы удовлетворить потребности более широкого круга пользователей.
2. Улучшение языковой поддержки. В рамках развития приложения планируется добавить поддержку множества языков, что сделает его доступным для более широкой аудитории, включая пользователей, говорящих на других языках.
3. Автоматизация процесса одобрения кредитов. В будущем будет реализована система автоматической оценки заявок на кредиты на основе анализа данных, таких как кредитная история, доход, возраст и другие показатели с использованием машинного обучения и нейронных сетей.
4. Мобильное приложение. В дальнейшем будет разработано мобильное приложение для платформ iOS и Android, которое обеспечит пользователям возможность управления заявками и кредитами на ходу.
5. Интерфейс для консультаций и обратной связи с клиентами. В приложение будет добавлена возможность для пользователей общаться с банком через чат-ботов или консультантов, а также предоставлена форма обратной связи для улучшения клиентского сервиса и получения отзывов.
6. Повышение безопасности и защита данных. В будущем будет внедрена поддержка двухфакторной аутентификации для пользователей и сотрудников, а также улучшены алгоритмы шифрования данных для повышения безопасности хранения и передачи информации.

Эти улучшения позволят значительно расширить функциональность веб-приложения "Scam Bank": улучшить качество обслуживания клиентов, обеспечив более гибкое и персонализированное взаимодействие.

## **6. ПРИЛОЖЕНИЯ**

### **6.1. Документация по сборке и разворачиванию приложения.**

1. Склонировать репозиторий с проектом [1] по ссылке: <https://github.com/moevm/nosql2h24-loans>.
2. Перейти в корневую директорию проекта.
3. Собрать приложение с помощью команды: `docker-compose build --no-cache`.
4. Запустить приложение с помощью команды: `docker-compose up`.
5. Открыть приложение в браузере по адресу `http://127.0.0.1:8080` или нажать на порт контейнера `frontend` в приложении Docker Desktop.

### **6.2. Инструкция для пользователя.**

- **Регистрация**

Для регистрации в системе перейдите на страницу авторизации и нажмите кнопку "Регистрация". Заполните все обязательные поля: имя, фамилия, email, пароль и подтверждение пароля. Нажмите кнопку "Регистрация". Если все данные введены корректно, вы получите сообщение об успешной регистрации.

- **Вход в систему**

Для входа в систему перейдите на страницу авторизации, введите ваш email и пароль, затем нажмите кнопку "Вход". Если данные введены корректно, вы будете перенаправлены на главную страницу.

- **Заполнение профиля клиента.**

Нажмите на свое имя на панели быстрого доступа. Для заполнения профиля клиента нажмите на кнопку "Редактировать" и заполните все необходимые поля: ФИО, пол, возраст, доход, статус самозанятости, место работы, семейное

положение, место работы супруга, доход супруга (если есть), наличие детей, паспортные данные серия/номер, имущество в собственности, контактный телефон, электронная почта. Нажмите кнопку "Сохранить".

- Заполнение профиля сотрудника.

Нажмите на свое имя на панели быстрого доступа. Для заполнения профиля сотрудника нажмите на кнопку "Редактировать" и заполните все необходимые поля: ФИО, пол, возраст, контактный телефон, электронная почта, паспортные данные серия/номер. Нажмите кнопку "Сохранить".

- Оформление кредита

Перейдите на главную страницу. Найдите интересующий вас кредит и нажмите на него. Перейдите на страницу оформления кредита. Заполните поля: сумма кредита, срок кредита, созаемщик (опционально), залог. Нажмите кнопку "Оформить". Если все данные введены корректно, ваша заявка будет отправлена на рассмотрение.

- Просмотр заявок

Перейдите на главную страницу. Нажмите кнопку "Заявки" на панели быстрого доступа. Просмотрите список ваших заявок, включая статус каждой заявки. Используйте кнопки сортировки и фильтрации для нахождения нужной заявки.

- Управление заявками

Для управления заявками сотрудником просмотрите список всех заявок клиентов. Используйте кнопки "Галочка" или "Крестик" для одобрения или отклонения заявок. Используйте кнопки сортировки и фильтрации для нахождения нужной заявки.

- Просмотр активных кредитов

Перейдите на главную страницу. Нажмите кнопку "Кредиты" на панели быстрого доступа. Просмотрите список всех открытых кредитов. Используйте кнопки сортировки и фильтрации для нахождения нужного кредита.

- Просмотр кредитной истории пользователя сотрудником

Для просмотра кредитной истории сотрудником найдите интересующего вас клиента и перейдите к его кредитной истории. Просмотрите список всех открытых и закрытых кредитов клиента. Используйте кнопки сортировки и фильтрации для нахождения нужного кредита.

- Просмотр статистики

Перейдите на главную страницу. Нажмите кнопку "Статистика" на панели быстрого доступа. Выберите параметры для построения статистики: тип кредита, параметр для статистики. Нажмите кнопку "Построить" для отображения статистики.

- Импорт/экспорт данных

Перейдите на страницу статистики. Нажмите кнопку "Импортировать". Выберите файл для загрузки в поддерживаемом формате (.json). Нажмите кнопку "Загрузить". Для экспорта данных нажмите кнопку "Экспортировать". Данные будут выгружены в файл с информацией о клиентах и их кредитах.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ссылка на код проекта на GitHub. – [Электронный ресурс]. – URL: <https://github.com/moevm/nosql2h24-loans>
2. Документация Python. – [Электронный ресурс]. – URL: <https://docs.python.org/3/> (дата обращения 22.12.2024).
3. Документация MongoDB. – [Электронный ресурс]. – URL: <https://www.mongodb.com/?msockid=06c9d97bda6161092c57ca3cdba160a5> (дата обращения 22.12.2024).
4. Документация pymongo. – [Электронный ресурс]. – URL: <https://pymongo.readthedocs.io/en/stable/> (дата обращения 22.12.2024).
5. Документация Vue.js. – [Электронный ресурс]. – URL: <https://vuejs.org/> (дата обращения 22.12.2024).
6. Документация Vue-chart.js. – [Электронный ресурс]. – URL: <https://vue-chartjs.org/migration-guides/> (дата обращения 22.12.2024).
7. Документация flask. – [Электронный ресурс]. – URL: <https://flask.palletsprojects.com/en/latest/> (дата обращения 22.12.2024).
8. Документация mongoengine. – [Электронный ресурс]. – URL: <https://mongoengine.readthedocs.io/en/latest/> (дата обращения 22.12.2024).
9. Документация pyjwt. – [Электронный ресурс]. – URL: <https://pyjwt.readthedocs.io/en/stable/> (дата обращения 22.12.2024).
10. Документация flask-cors. – [Электронный ресурс]. – URL: <https://flask-cors.readthedocs.io/en/latest/> (дата обращения 22.12.2024).
11. Документация datetime. – [Электронный ресурс]. – URL: <https://datetime.readthedocs.io/en/stable/> (дата обращения 22.12.2024).
12. Документация numpy. – [Электронный ресурс]. – URL: <https://numpy.org/doc/stable/> (дата обращения 22.12.2024).
13. Документация python-dateutil. – [Электронный ресурс]. – URL: <https://dateutil.readthedocs.io/en/stable/> (дата обращения 22.12.2024).