ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»**

Факультет математики и информационных технологий

Кафедра фундаментальной информатики

УТВЕРЖДАЮ

Зав. кафедрой
канд. физ.-мат. наук, доцент

_____ А. Г. Смольянов
(подпись)

«15» 06 2018 г.

БАКАЛАВРСКАЯ РАБОТА

РАЗРАБОТКА ВЕБ-СЕРВИСА ДЛЯ РАСЧЕТА ФИНАНСОВЫХ ПОКАЗАТЕЛЕЙ

Автор бакалаврской работы (подпись) 13.06.2018 (дата) С. И. Абрамов

Обозначение бакалаврской работы БР–02069964–02.03.02 –01–18

Направление 02.03.02 Фундаментальная информатика и информационные технологии

Руководитель работы
канд. физ.-мат. наук (подпись) 15.06.2018 (дата) А. В. Попов

Нормоконтролер
канд. пед. наук, доцент (подпись) 15.06.2018 (дата) И. А. Карпюк

Рецензент
канд. физ.-мат. наук,
зав. каф. алгебры и геом. (подпись) 15.06.2018 (дата) Л. А. Сухарев

Саранск
2018

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. Н. П. ОГАРЁВА»

Факультет математики и информационных технологий

Кафедра фундаментальной информатики

УТВЕРЖДАЮ

Зав. кафедрой
канд. физ.-мат. наук, доцент

_____ А. Г. Смольянов
(подпись)

«29» 12 2017 г.

ЗАДАНИЕ НА ВЫПУСНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Студент Абрамов Сергей Игоревич

1 Тема «Разработка веб-сервиса для расчета финансовых показателей»

Утверждена приказом № 10322-с от 29.12.2017 г.

2 Срок предоставления работы к защите 01.06.2018

3 Исходные данные для научного исследования литература и электронные источники по вопросам разработки клиент-серверных приложений

4 Содержание бакалаврской работы

4.1 Проектирование веб-сервиса

4.2 Технологические решения для разработки веб-сервиса

4.3 Разработка веб-сервиса

5 Приложения Код функций для авторизации и регистрации пользователей

Руководитель работы
канд. физ.-мат. наук _____ (подпись) 29.12.2017 (дата) А. В. Попов

Задание принял к исполнению _____ (подпись) 29.12.2017 (дата) С. И. Абрамов

РЕФЕРАТ

Бакалаврская работа содержит 103 страницы, 45 рисунков, 15 использованных источников, 1 приложение.

PHP, MYSQL, JQUERY, JAVASCRIPT, AJAX, HTML, BOOTSTRAP, ИНТЕРФЕЙС, КРЕДИТ, ВЕБ-СЕРВИС, ИНТЕРНЕТ, СЕРВЕР, РЕЛЯЦИОННАЯ БАЗА ДАННЫХ, СВЯЗИ, ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ.

Объектом исследования является система управления базами данных MySQL, веб-технологии PHP, HTML, JavaScript, jQuery, AJAX и Bootstrap в целях разработки веб-сервиса для расчета финансовых показателей.

Цель работы – рассмотрение существующих технологий для создания клиент-серверных приложений, проектирование и реализация базы данных веб-сервиса, создание веб-интерфейса и механизма авторизации и регистрации пользователей, реализация функций для работы с возможностями сервиса.

В результате проведенной работы были изучены технологии, применяемые в настоящее время для реализации клиент-серверных приложений, проанализированы особенности и возможности популярных клиентских и серверных языков программирования, систем управления базами данных, способов размещения приложений в сети Интернет. Спроектирована и разработана база данных, реализован веб-интерфейс для работы с веб-сервисом, создан механизм аутентификации пользователей, реализованы функции для работы с возможностями сервиса.

Степень внедрения – база данных, веб-интерфейс и скрипты размещены на хостинге.

Эффективность – облегчение процесса расчета графиков погашения кредита, снижение трудозатрат при вычислении максимальной суммы кредита, которая с большой долей вероятности может быть выдана организации банком, что позволит получить максимальную выгоду от использования кредита.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 6
1 Проектирование веб-сервиса 8
1.1 Исследование предметной области 8
1.1.1 Кредит 8
1.1.2 Актуальность веб-сервиса для расчета финансовых показателей 9
1.2 Назначение и основные возможности проектируемого веб-сервиса 10
1.3 Общая архитектура веб-сервиса 11
1.4 Проектирование серверной части 11
1.5 Проектирование базы данных 12
1.5.1 Анализ предметной области 12
1.5.2 Концептуальное проектирование базы данных 14
1.5.3 Логическое и физическое проектирование базы данных 15
1.6 Проектирование клиентской части 19
2 Технологические решения для разработки веб-сервиса 20
2.1 Выбор способа реализации серверной части 20
2.2 Выбор языков программирования для разработки веб-сервиса 20
2.3 Требования к клиентской части 21
3 Разработка веб-сервиса 22
3.1 Разработка интерфейса клиентской части. 22
3.1.1 Требования к дизайну веб-сервиса 22
3.1.2 Структура клиентской части 23
3.1.3 Front-end фреймворк Bootstrap 24
3.1.4 Верстка веб-сервиса 26
3.2 Реализация кредитного онлайн калькулятора 28
3.2.1 Верстка интерфейса калькуляторов 28
3.2.2 Разработка с использованием библиотеки jQuery 30
3.2.3 Реализация алгоритмов расчёта графиков платежей. 34
3.2.4 Сохранение графиков в pdf 37

3.2.5 Сохранение графика платежей в xls 39
3.3 Реализация базы данных 39
3.4 Реализация авторизации и регистрации на сервисе 40
3.4.1 Авторизация через Вконтакте 53
3.4.2 Авторизация через Telegram 57
3.4.3 Авторизация через Google 61
3.5 Реализация расчета кредитного лимита 67
ЗАКЛЮЧЕНИЕ 74
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 76
ПРИЛОЖЕНИЕ А (обязательное) Функции для авторизации и регистрации пользователей 78

ВВЕДЕНИЕ

Современную экономическую жизнь невозможно представить без кредитных отношений. Благодаря кредитам, у юридических и физических лиц существует возможность решать финансовые проблемы. Для физических лиц кредит служит инструментом для приобретения дорогостоящих вещей, жилья и многообразия представленных на рынке товаров и услуг. У юридических лиц сфера применения кредитов гораздо шире. Благодаря кредитованию многие начинающие предприниматели могут организовать собственное дело, а опытные бизнесмены – увеличить эффективность деятельности фирмы.

Однако для получения максимальной выгоды требуется, с одной стороны, заранее рассчитать график погашения кредита с учетом особенностей программы займа, а с другой стороны – предварительно оценить максимальную сумму кредита, которую с большой долей вероятности одобрит банк. Расчет графика погашения кредита позволяет компании правильно оценить свои возможности, для погашения кредита, а также выбрать для себя наиболее удобную схему погашения кредита. В настоящее время каждый банк при выдаче кредита проверяет заемщика в бюро кредитных историй. В случае просроченных платежей или отказа банком в выдаче кредита, кредитная история организации может быть ухудшена. Отказ банка также приведет к повторной подготовке множества необходимых документов, что очень трудозатратно и требует много времени. Оценка максимальной суммы кредита направлена именно на предотвращение отказа банков в выдаче кредитов, что позволит организации избежать описанных проблем.

Подобные задачи можно выполнить вручную, но это сложно, времязатратно и легко допустить ошибку в вычислениях. Поэтому намного удобнее и надежнее выполнить эти задачи при помощи специально разработанного под эти цели сервиса.

Для решения задач по расчёту графиков погашения кредита и оценки максимальной суммы кредита необходимо использование системы, состоящей

из базы данных и интерфейса для работы с этой базой. Поскольку сервис рассчитан под использование различными организациями, то сервис должен быть кроссплатформенным, то есть не зависеть от операционной системы пользователя. Наиболее доступной средой для подобных сервисов является Интернет. Единственным условием для работы с такой системой является наличие устойчивого соединения с сетью Интернет.

Целью данной бакалаврской работы является разработка веб-сервиса, позволяющего в онлайн режиме рассчитать график погашения кредита по различным схемам платежей, а также на основании различных финансовых показателей провести предварительный расчет суммы кредита, который с большой долей вероятности может быть выдан определенной компании. В соответствии с целью были сформулированы следующие задачи:

- провести анализ предметной области (процесс взаимодействия компаний и предпринимателей с кредитными организациями), выполнить общее проектирование веб-сервиса;

- рассмотреть современные технологии, используемые в настоящее время для быстрой разработки веб-приложений;

- разработать несколько функциональных модулей создаваемого веб-сервиса.

Бакалаврская работа состоит из введения, трех разделов и заключения.

В первом разделе дана постановка задачи создания сервиса для ввода и хранения финансовых показателей, описан процесс проектирования этого веб-сервиса, состоящего из базы данных, серверной и клиентской части.

Во втором разделе рассмотрены современные технологии для создания клиент-серверных приложений, выбраны языки и фреймворки, которые будут использованы в дальнейшем при разработке.

В третьем разделе описан процесс практической разработки функциональных модулей веб-сервиса, предназначенных для авторизации и аутентификации пользователей на сайте, расчета и визуализации кредитного калькулятора, ввода и сохранения данных о группах связанных заемщиков.

1 Проектирование веб-сервиса

Первый этап в создании продукта – проектирование. Это этап цикла программного продукта, во время которого исследуется структура и взаимосвязи элементов разрабатываемого веб-сервиса. Проектирование позволяет структурировать идеи, избежать ошибки и избавиться от лишней работы на ранних стадиях разработки. В свою очередь, это ускорит процесс разработки веб-сервиса.

1.1 Исследование предметной области

1.1.1 Кредит

Кредит – экономические отношения, при которых одна из сторон не возмещает полученные от другой стороны деньги немедленно, но обещает вернуть или возместить ресурсы в будущем. Кредит – юридическое оформление экономического обязательства [12].

В практической экономической деятельности кредит представляет собой передачу форме во временное использование материальных ценностей в денежной либо товарной форме. Кредитные отношения проявляются в виде конкретных кредитных сделок, формы и условия которых отличаются значительным разнообразием. Суть кредита всегда неизменна. Он сохраняет черты, свойственные экономическим отношениям, лежащим в основе кредита.

В любой кредитной сделке участвуют два субъекта – кредитор и заемщик.

Кредитор – это субъект кредитного отношения, предоставляющий ссуду заемщику во временное использование. Кредиторами в настоящее время могут выступать не только банки, но различные предприятия, частные фирмы, государство, инвестиционные фонды, физические лица и многие другие

Заемщик – это субъект кредитного отношения, получающий во временное использование ссуду и обязанный ее вернуть в установленный срок кредитным

договором срок. Заемщиками могут быть как юридические, так и физические лица, испытывающие потребность в денежных средствах.

Кроме кредиторов и заемщиков, элементом кредитной сделки является объект передачи. Это то, что собственно и передается от кредитора к заемщику, а потом совершает обратный путь от заемщика к кредитору. Объектом передачи выступает ссуженная стоимость.

1.1.2 Актуальность веб-сервиса для расчета финансовых показателей

В России по данным статистики Центрального банка Российской Федерации за 2017 год юридическим лицам было выдано 209 430 639 млн. рублей. Эта сумма хорошо отображает значимость кредитов для юридических лиц. Кредит, в первую очередь, – это инвестиция для развития бизнеса. Начало своего бизнеса с нуля, его развитие и расширение, модернизация оборудование – все это очень затратные операции. И грамотное распоряжение кредитными отношениями может сильно поднять уровень компании на рынке, обойти конкуренцию, а в трудных ситуациях избежать банкротства.

Необходимость кредитования юридических лиц обусловлена коммерческой организацией управления в условиях рынка, когда на каждом предприятии в условиях оборота капитала появляется дополнительная потребность в средствах. При помощи кредитных отношений организации получают необходимые для нормальной работы средства.

Но огромное разнообразие форм и условий кредитования может ужаснуть человека, не связанного с сферой кредитных отношений. В результате нужно потратить слишком много времени на изучение различных видов кредитования, формул расчета. Чаще всего организации приходится даже нанять сотрудника, который будет работать только в сфере кредитования. Это влечет дополнительные расходы денежных средств, которых чаще всего и так не хватает.

Создание веб-сервиса для расчета финансовых показателей поможет сделать взаимодействие с кредитами более удобным. Любой сотрудник компании, имеющий доступ в сеть Интернет и знающий необходимые сведения о компании, может посетить сервис и за несколько минут получить готовые графики погашения кредита, либо узнать максимальную сумму кредита, которую может получить его организация. Пропадает необходимость вникать в многочисленные формулы, изучать алгоритмы по вычислению кредитных лимитов. Сотрудники экономят время и могут его потратить на выполнение других задач. Кроме того, сервис позволяет быстро провести расчёты по разным схемам платежей, с различными сроками и размерами процентной ставки, что помогает сравнить полученные результаты и определить, какой кредит выгоднее взять организации или предпринимателю.

1.2 Назначение и основные возможности проектируемого веб-сервиса

Веб-сервис создается для обслуживания компаний и индивидуальных предпринимателей, обращающихся в банки для получения кредита. Сервис должен помочь понять, на какую сумму кредита может рассчитывать организация или предприниматель.

Основные составляющие создаваемого веб-сервиса:

- Расчет графика погашения кредита по различным схемам платежей.

- Предварительный расчет суммы кредита, который может быть выдан определенной компании.

Проект имеет потенциал в сфере кредитования, так как объемы кредитования юридических лиц растут с каждым годом.

1.3 Общая архитектура веб-сервиса

Архитектура информационной системы – это концепция, которая определяет модель, выполняемые функции, структуру и взаимосвязь компонентов этой системы.

Наш проект будет реализован в виде распределенной системы с архитектурой «клиент-сервер». Взаимодействие клиентов и сервера будет происходить в среде Интернет (рисунок 1.1).

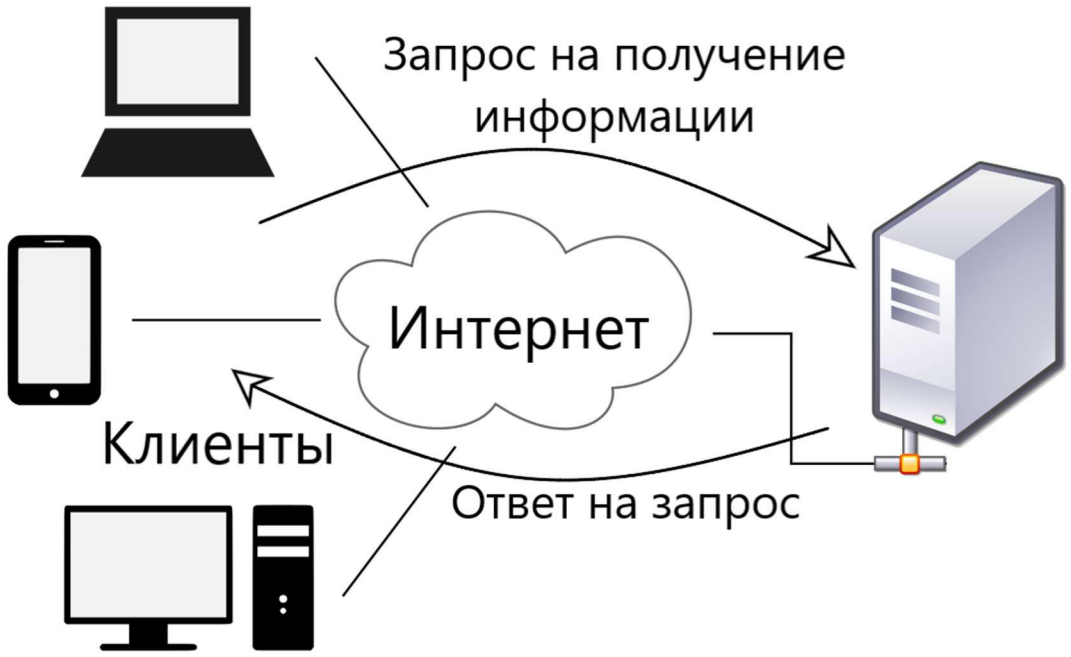


Рисунок 1.1 – Общая архитектура веб-сервиса.

1.4 Проектирование серверной части

Расчет максимальной суммы кредита производится на основании множества финансовых показателей организации, которые необходимо постоянно хранить. Поэтому в серверную часть веб-сервиса необходимо добавить технологии работы с базами данных. Детализация архитектуры серверной части веб-сервиса изображена на рисунке 1.2.

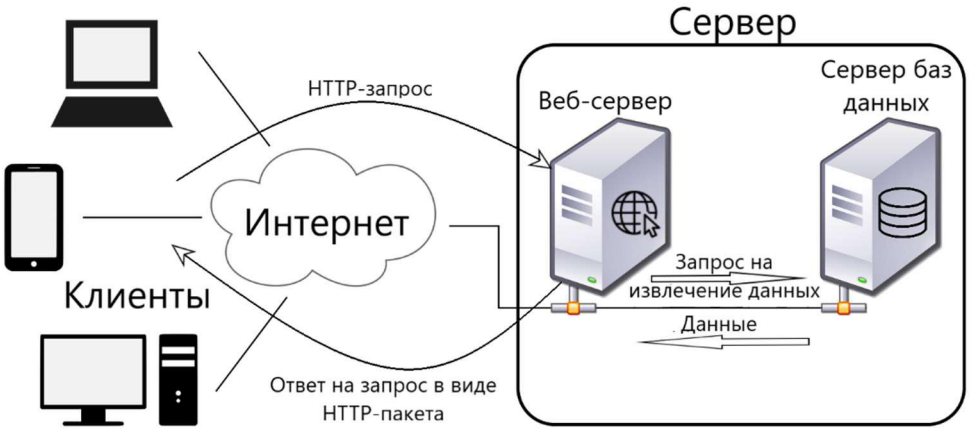


Рисунок 1.2 – Детализация серверной части.

1.5 Проектирование базы данных

Процесс проектирования базы данных для разрабатываемого сервиса состоит из нескольких этапов.

1.5.1 Анализ предметной области

При расчете суммы кредита, на которую может рассчитывать компания, учитываются различные экономические и финансовые параметры. Все эти параметры, а также все введенные пользователями данные о компаниях следует хранить в базе данных. Сведения о компании характеризуются следующей информацией:

1) Название компании.

2) Индивидуальный номер налогоплательщика.

3) Дата регистрации компании.

4) Дата начала работы компании.

5) Группа связанных заемщиков, к которой относится эта организация.

6) Используемая система налогообложения.

7) Организационно-правовая форма компании.

Существует возможность категорировать компании, по группам связанных заемщиков, по системе налогообложения и организационно-

правовой форме. Информация о каждой группе связанных заемщиков содержит в себе название группы и её описание.

Каждый вид системы налогообложения характеризуется следующей информацией:

1) Название вида налогообложения.

2) Описание конкретного вида.

3) Признак, указывающий на то, используется ли компания при расчете суммы кредита, или нет.

Информация об организационно-правовых формах (ОПФ) содержит в себе:

1) Название организационно-правовой формы.

2) Описание конкретной формы.

3) Количество знаков в индивидуальном номере налогоплательщика (ИНН), имеющего данную ОПФ.

4) Признак того, что компания является организацией, а не индивидуальным предпринимателем.

Для расчета максимального лимита по кредиту, необходимо реализовать хранение финансовых данных, составляющих различным статьям баланса организации:

1) Внеоборотные активы.

2) Оборотные активы.

3) Капитал и резервы.

4) Долгосрочные обязательства.

5) Краткосрочные обязательства.

Для предотвращения компрометации данных о компаниях необходимо реализовать механизмы авторизации и аутентификации пользователей. Информация о пользователе включает в себя:

1) Фамилию, имя и отчество конкретного пользователя.

2) Адрес электронной почты.

3) Псевдоним, заданный пользователем.

4) Параметр, указывающий на согласие пользователя о получении информационных сообщений от администрации.

5) Признак того, что пользователь подтвердил введенный адрес электронной почты.

Вследствие этого, в информации о компании и группе связанных заемщиков, так же должны присутствовать сведения о том, какому из пользователей они принадлежат.

Для авторизации через социальные сети необходимо хранение данных социальных сетей. Информация о социальных сетях содержит:

1) Идентификатор Вконтакте.

2) Идентификатор Телеграмм.

3) Идентификатор Google.

Дополнительно в системе необходимо хранить служебную информацию (статьи баланса предприятий, даты расчета лимитов, временные данные при восстановлении пароля или регистрации через социальные сети).

1.5.2 Концептуальное проектирование базы данных

Цель концептуального проектирования – это создание модели данных на основе анализа предметной области. Наиболее часто используемой моделью данных является ER-модель (модель сущность-связь). Это модель данных, которая позволяет описать предметную область в концептуальных схемах. ER-модель имеет разные графические средства визуализации, но в качестве стандартного средства используется ER-диаграмма. Такая диаграмма в наглядном виде демонстрирует связь между сущностями [1].

ER-диаграмма проектируемой базы данных изображена на рисунке 1.3.

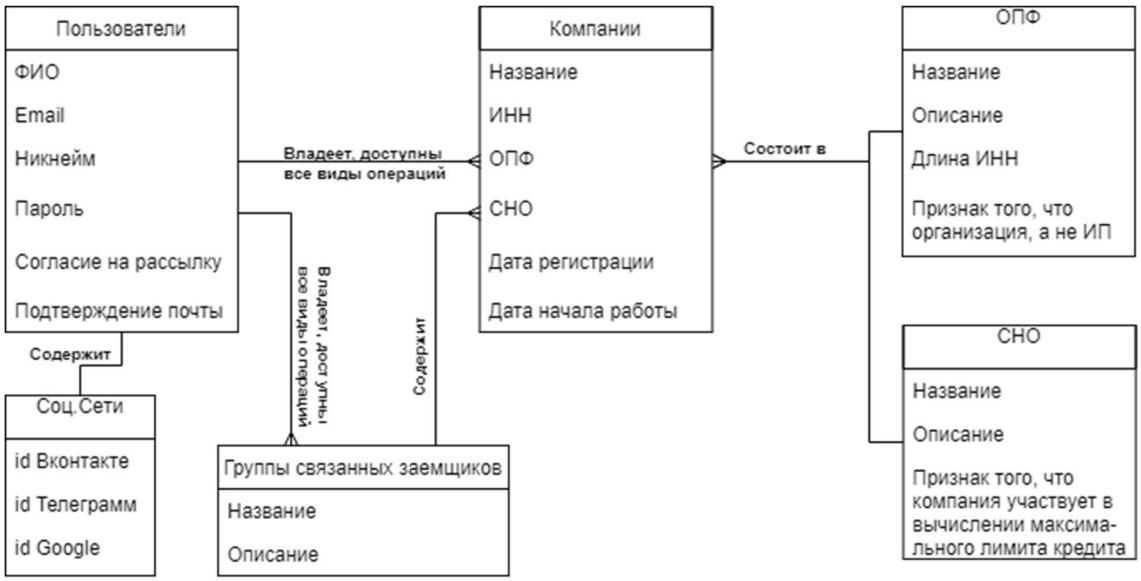


Рисунок 1.3 – Концептуальная модель.

Далее полученную концептуальную модель нужно преобразовать в логическую модель данных, состоящую из таблиц, полей, ключей и т.п.

1.5.3 Логическое и физическое проектирование базы данных

Логическая (реляционная) модель данных нашего сервера изображена на рисунке 1.4

Обозначения в модели:

- PK (Primary key) – первичный ключ. Это столбец или комбинация столбцов, позволяющих однозначно идентифицировать запись.

- FK (Foreign key) – внешний ключ. Это столбец или комбинация столбцов в дочерней таблице, соответствующих столбцу или комбинации столбцов из родительской таблицы, представленных там как первичный ключ.

- U1, U2, U3 (Unique) – уникальный ключ. поля или комбинации поля, обозначенные данным ключем, должны быть уникальными.

- UNSIGNED – атрибут, обозначающий, что поле не может принимать отрицательные значения.

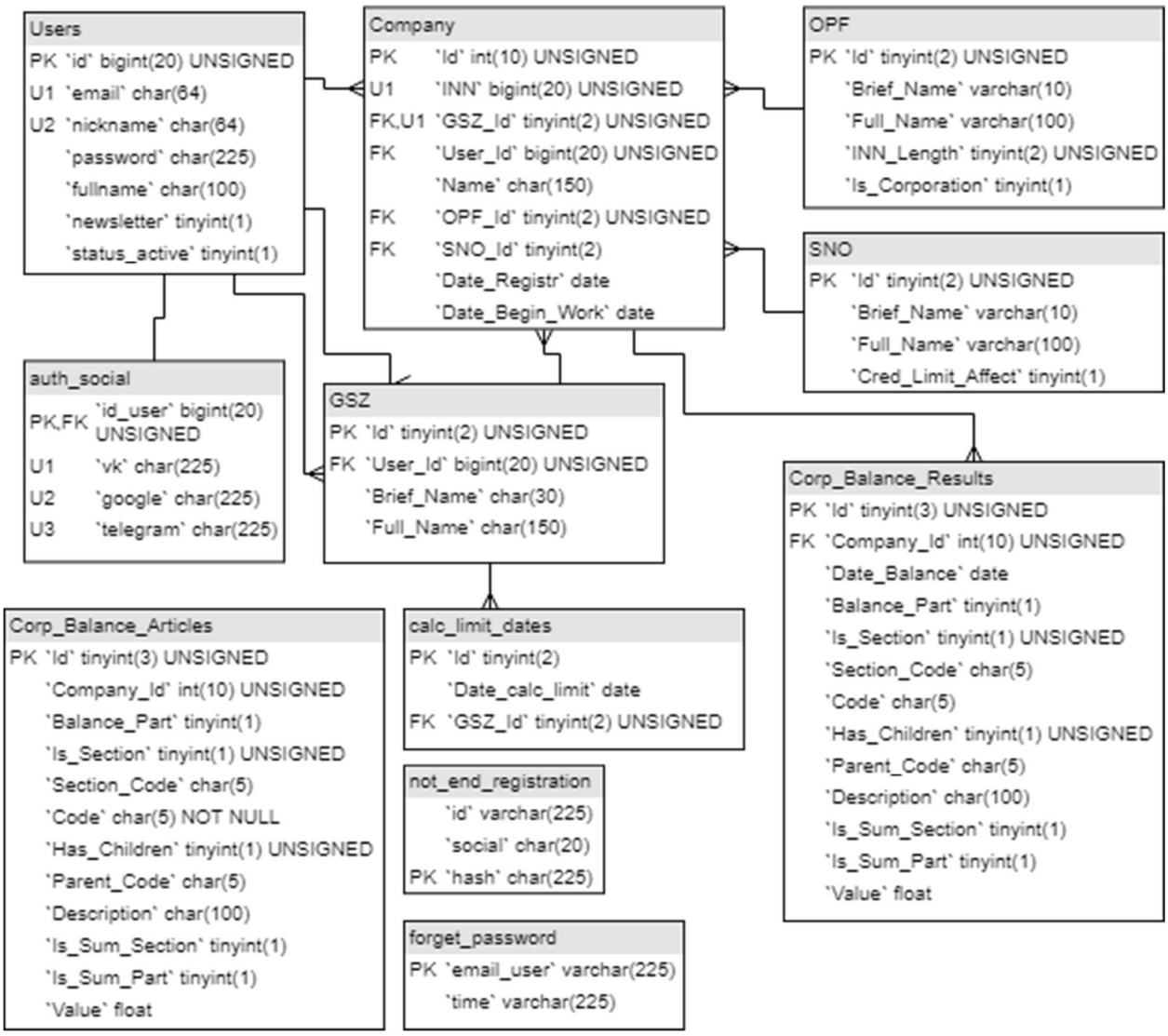


Рисунок 1.4 – Логическая модель данных

Рассмотрим поля в каждой используемой таблице.

users – информация о пользователях:

- *id* – идентификатор пользователя;
- *email* – адрес электронной почты;
- *nickname* – псевдоним пользователя;
- *password* – хеш пароля пользователя;
- *fullname* – ФИО пользователя;
- *newsletter* – согласие пользователя на рассылку;
- *status_active* – статус подтверждения почты.

company – информация о компаниях:

- *id* – идентификатор компании;

- *INN* – идентификационный номер налогоплательщика;

- *GSZ_id* – id группы связанных заемщиков из таблицы GSZ;

- *User_id* – id пользователя, добавившего компанию, из таблицы users;

- *Name* – название компании;

- *OPF_id* – id организационно-правовой формы из таблицы OPF;

- *SNO_id* – id системы налогообложения из таблицы SNO;

- *Date_Registr* – дата регистрации компании;

- *Date_Begin_Work* – дата начала работы компании.

GSZ –информация о группах связанных заемщиков:

- *id* – идентификатор компании;

- *User_id* – id пользователя, добавившего компанию, из таблицы users;

- *Brief_Name* – название группы;

- *Full_Name* – описание группы;

OPF – справочник видов организационно-правовых форм:

- *id* – идентификатор формы;

- *Brief_Name* – название;

- *Full_Name* – описание;

- *INN_Length* – длина ИНН для данной ОПФ;

- *Is_Corporation* – признак того, что компания является организацией, а не индивидуальным предпринимателем.

SNO – справочник видов налогообложения:

- *id* – идентификатор вида;

- *Brief_Name* – название;

- *Full_Name* – описание;

- *Cred_Limit_Affect* – признак, указывающий на то, используется ли компания при расчете суммы кредита.

auth_social – идентификаторы социальных сетей пользователей:

- *id_user* – id пользователя, добавившего компанию, из таблицы users;

- *vk* – идентификатор Вконтакте;

- *telegram* – идентификатор Телеграмм;

- *google* – идентификатор Google.

calc_limit_dates – даты расчета максимального лимита кредита:

- *id* – идентификатор даты;

- *Date_calc_limit* – дата расчета лимита кредита;

- *GSZ_id* – id группы связанных заемщиков из таблицы GSZ.

Corp_Balance_Results – баланс организации:

- *id* – идентификатор заголовка;

- *Company_Id* – id компании;

- Date_Balance – дата баланса;

- *Balance_Part* – признак, указывающий на дебит(1) и кредит(2);

- *Is_Section* – признак того, что строка является разделом баланса.

Section_Code – код раздела:

- *Code*– код статьи;

- *Has_Children*– признак, указывающий существуют ли подкоды;

- *Parent_Code*– код статьи родителя;

- *Description*– описание статьи;

- *Is_Sum_Section*– если 1, то сумма раздела;

- *Is_Sum_Part*– если 1, то сумма актива или пассива;

- *Value*– сумма по статье баланса.

Таблица ***Corp_Balance_Articles*** аналогична таблице ***Corp_Balance_Results****,* за исключением того, что в ней хранятся только заголовки статей, а не сами значения.

forget_password – временные данные при восстановление пароля:

- *email_user* – адрес электронной почты пользователя, запросившего пароль;

- *time* – время в формате microtime, служит для очистки таблицы от неактуальной информации и проверки хеша из массива.

not_end_registration – временные данные, для пользователей, которые пробовали зайти через социальные сети, прежде чем зарегистрироваться:

- *id* – идентификатор учетной записи пользователя в социальной сети;

- *social* – название социальной сети, через которую заходил пользователь;

- *hash* – хеш, для проверки того, что во время регистрации информация о социальной сети и id не были повреждены или изменены специально.

Цель физического проектирования – это описание способа физической реализации полученной логической (в нашем случае реляционной) модели базы данных. Необходимо создать набор реляционных таблиц, определить индексы, связи и ограничения для полей, а также выбрать определенную систему хранения данных для обеспечения оптимальной производительности СУБД.

В нашем проекте будет использована система хранения данных InnoDB СУБД MySQL. Основным отличием ее от других подсистем, таких как MyISAM, является наличие внешних ключей и поддержка транзакций [10]. За счёт этого данная система хранения может обеспечить лучшую целостность данных. Система хорошо ведет себя при смешанной нагрузке, что хорошо скажется на скорости работы с базой данных в планируемых сценариях использования.

1.6 Проектирование клиентской части

Клиентская часть разрабатываемого веб-сервиса должна работать с помощью стандартных современных браузеров Google Chrome, Mozilla Firefox или Microsoft Edge без установки дополнительных модулей или библиотек.

2 Технологические решения для разработки веб-сервиса

Вторым этапом в разработке веб-сервиса будет выбор наиболее оптимальных существующих технических средств в соответствии с результатом проектирования из первой главы.

2.1 Выбор способа реализации серверной части

Перед разработкой проекта нужно решить, на какой платформе он будет работать. Разрабатываемый веб-сервис не требует больших мощностей и может функционировать на виртуальном хостинге (без аренды выделенного виртуального или физического сервера) и на самой распространенной бесплатной реляционной СУБД MySQL.

2.2 Выбор языков программирования для разработки веб-сервиса

Архитектура разрабатываемого веб сервиса состоит из двух частей: серверной и клиентской. Перед реализацией необходимо выбрать, какие языки будут использоваться на серверной части, а какие на клиентской.

Серверные языки программирования нужны для реализации бизнес-логики разрабатываемого продукта. Таких языков довольно много: PHP, Java, Ruby, Python, C и другие. Однако, для большинства проектов выбор языка программирования имеет мало значения и определяется предпочтениями разработчика. Для серверной части разрабатываемого продукта будет использоваться язык с открытым исходным кодом – PHP. Это широко используемый скриптовый язык общего назначения. Аббревиатура PHP означает PHP: Hypertext Preprocessor – «PHP: препроцессор гипертекста» [5]. Основными преимуществами этого языка программирования является:

- предоставление разработчикам возможности создания динамически генерируемых веб-страниц;

- традиционность языка и его простота, многие конструкции позаимствованы из других языков;

- PHP может быть встроен в код html страницы и будет корректно обработан интерпретатором;

- имеются гибкие настройки для обеспечения безопасности;

- в стандартный набор входит ряд механизмов шифрования;

- бесплатное распространение.

PHP позволяет за короткие сроки создавать качественные продукты, которые будет легко модифицировать и поддерживать в будущем. Дистрибутив PHP доступен для большинства операционных систем и поддерживается всеми современными хостинг-провайдерами [6].

Клиентская часть проекта будет реализована с помощью стандартных языков и инструментов, поддерживаемых веб-браузерами: HTML (HyperText Markup Language) для определения структуры страниц сайта, CSS (Cascading Style Sheets) для задания их дизайна и JavaScript для обеспечения динамической интерактивности.

2.3 Требования к клиентской части

Для оптимального использования разрабатываемого веб-сервиса клиентская часть должна соответствовать следующим требованиям:

- браузер, поддерживающий работу XHTML 1.0 и выше и сценарии JavaScript;

- доступ к сети Интернет со скоростью более 256 Кбит\сек.

Низкие требования к клиентской части позволят использовать веб-сервис практически с любого устройства, имеющего доступ к сети Интернет.

3 Разработка веб-сервиса

После проектировки разрабатываемого сервиса и выбора технологических решений, которые будут использоваться в разработке, можно перейти непосредственно к разработке веб-сервиса. Для этого необходимо выполнить ряд задач, а именно:

- разработать интерфейс клиентской части сервиса;
- создать базу данных на основе разработанной логической модели данных;
- разработать функции, реализующие функции веб сервиса.

3.1 Разработка интерфейса клиентской части.

3.1.1 Требования к дизайну веб-сервиса

Необходимо реализовать максимально понятный, простой и легкий интерфейс для клиентов сервиса. Для удобства использования сервиса клиентами как с компьютеров, так и с мобильных устройств существует необходимость в создании адаптивного веб-дизайна.

Адаптивный веб-дизайн – это дизайн веб-страниц, способный обеспечить правильное отображение страниц на устройствах с различными размерами окна браузера. Основная цель такого дизайна – универсальность отображения страниц для различных устройств. При этом пропадает необходимость создания отдельных версий для каждого вида устройств, один сайт без проблем работает на всем спектре устройств.

В данный момент веб-сервисы, разработанные с использованием адаптивной верстки, выдаются поисковыми системами с большим приоритетом.

Разработанный дизайн должен самостоятельно изменять расположения и размеры окон, шрифтов и прочих элементов дизайна в зависимости от разрешения окна, где он отображается.

Главная страница сайта создает основное впечатление пользователя от ресурса. Она должна соответствовать следующим требованиям:

- точно отображать тематику сервиса;
- не содержать лишней и отвлекающей информации;
- должна быть понятной и удобной для пользователя.

Все остальные страницы сервиса должны иметь общий дизайн с главной страницей, но иметь уникальное содержание.

3.1.2 Структура клиентской части

Так как веб-сервис служит для расчета финансовых показателей, нужно сосредоточить внимание пользователя на основных функциях сервиса. С главной страницы должен быть обеспечен быстрый доступ к кредитному калькулятору и расчету суммы кредита. Для действующих пользователей должна быть возможность быстрой авторизации, а после авторизации – доступ к своим данным с любой страницы сайта.

Согласно описанным требованиям, на рисунке 3.1 представлена оптимальная структура для разрабатываемого веб-сервиса.

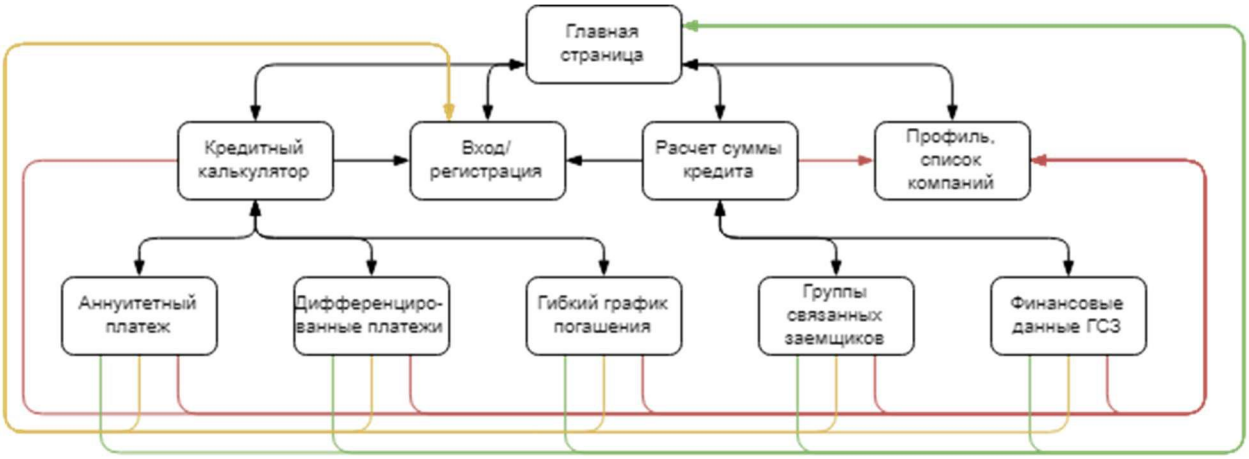


Рисунок 3.1 – Структура веб сервиса

3.1.3 Front-end фреймворк Bootstrap

Для реализации поставленных задач был выбран фреймворк Bootstrap. На данный момент это самый популярный HTML, CSS и JavaScript фреймворк для создания сайтов и веб-приложений. Основная цель его использования – сделать веб-разработку более эффективной и быстрой [3].

Если разрабатывать сайт с нуля, то придется продумывать множество вещей. Все веб-сценарии, css-стили придется писать самостоятельно, а часто они составляют тысячи строчек кода. Можно запросто совершить ошибку в верстке, из-за чего сервис будет по-разному выглядеть в разных браузерах.

Основные преимущества фреймворка Bootstrap:

- адаптивность, стандартизация интерфейсов, высокая скорость работы;
- единый стиль оформления всех страниц сайта;
- экономия времени при разработке;
- простота использования и изучения.

Bootstrap содержит набор HTML+CSS шаблонов и инструментов, а так же множество JS-функций [14]. На его основе можно сделать практически любой интерфейс без особых затрат по времени. Отдельное внимание стоит уделить адаптивности. По данным инструмента для анализа веб-трафика StatCounter в 2016 году произошел переломный момент – доля просмотра веб-сайтов с мобильных устройств превысила долю просмотра с настольных устройств. На момент написания данной работы доля мобильных устройств составляет 54.11%, а настольных 45.89%. Сравнение графиков долей мобильных и настольных устройств представлено на рисунке 3.2.

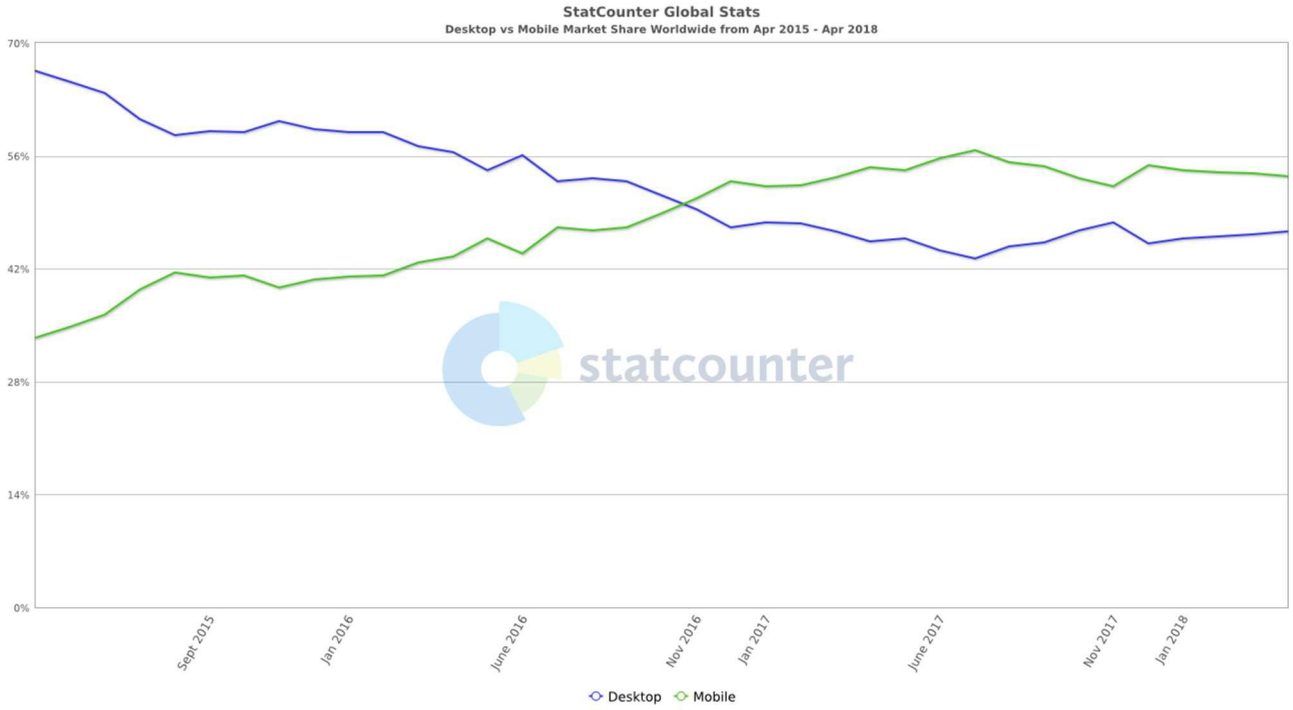


Рисунок 3.2 – Распределение долей просмотра веб-сайтов с мобильных и настольных устройств за 2015-2018 года

Это означает, что современный веб-разработчик должен думать о правильном функционировании сервиса не только на компьютерах, но и на мобильных устройствах. Можно, конечно, создать отдельную мобильную версию, но это выльется в отдельный проект и понесет за собой огромные затраты времени и денег.

В Bootstrap воплощена концепция отзывчивого веб-дизайна. Сайт отображает наиболее полную информацию вне зависимости от размера экрана устройства. На веб-сайте не меняется содержание и цветовая гамма, меняется только форма и способ группировки блоков сайта, с целью сделать использование веб-сервиса наиболее удобным для пользователя.

На данный момент самая свежая версия Bootstrap 4, но новая версия не успела перенять все возможности прошлой версии. Для третьей версии существует гораздо больше документации, что благоприятно скажется при разработке сервиса. Поэтому для разрабатываемого сервиса было решено использовать Bootstrap 3.

3.1.4 Верстка веб-сервиса

Определив требования к дизайну и разработав структуру веб-сервиса, можно приступить к созданию клиентской части.

Наш проект будет реализован в виде сайта, состоящего из нескольких HTML-страниц (Multi Page Application). Главная главная страница сайта index.html изображена на рисунке 3.3.

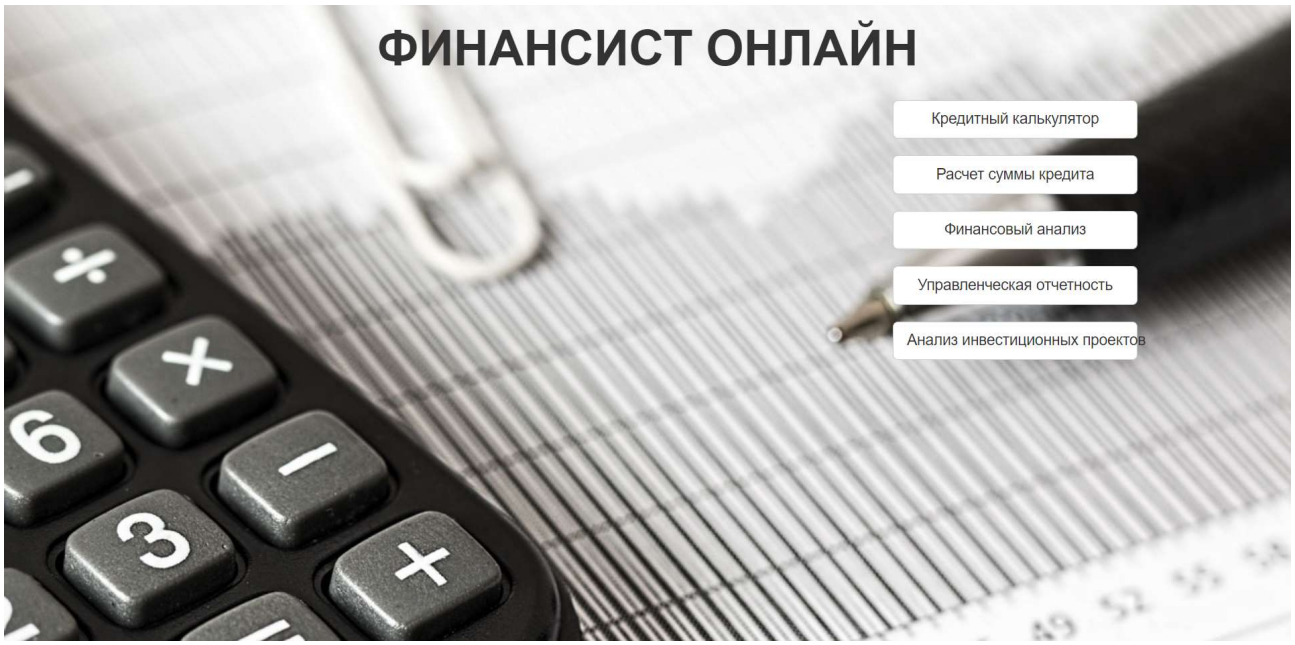


Рисунок 3.3 – Главная страница веб-сервиса

Разработанный дизайн не содержит ничего лишнего и максимально акцентирует внимание пользователя на возможностях сайта. Кнопки главного меню выполнены при помощи фреймворка Bootstrap. Код одной из кнопок:

```
«<a class="btn_margin btn btn-default btn-lg col-xs-10 col-sm-6 col-md-4
col-lg-3 col-xs-offset-1 col-sm-offset-5 col-md-offset-7 col-lg-offset-9"
href="./cred_calc/calc.html">Кредитный калькулятор </a>»
```

Как можно заметить, кнопка – это обычная ссылка. Все оформление этой кнопки задается с помощью перечисления в атрибуте class.

Используя Bootstrap, удалось добиться адаптивности дизайна. Вид главной страницы сервиса с мобильного устройства изображен на рисунке 3.4.

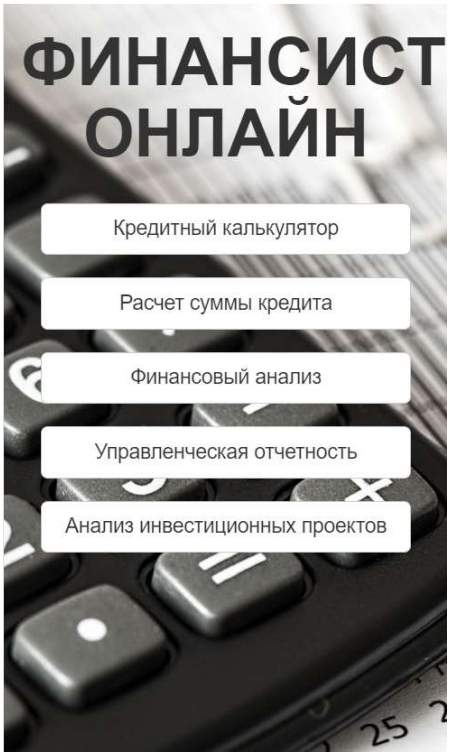


Рисунок 3.4 – Главная страница с мобильного устройства

Поскольку в дальнейшем будет реализована авторизация и регистрация пользователей, то необходимо наличие шапки сайта, позволяющей быстро авторизоваться или перейти на главную страницу с любой другой страницы, а для авторизованных пользователей содержащую ссылки на профиль и список компаний, а также выход из учетной записи. Разработанная шапка для неавторизованного пользователя представлена на рисунке 3.5. В случае если пользователь авторизован, то он видит шапку, представленную на рисунке 3.6.

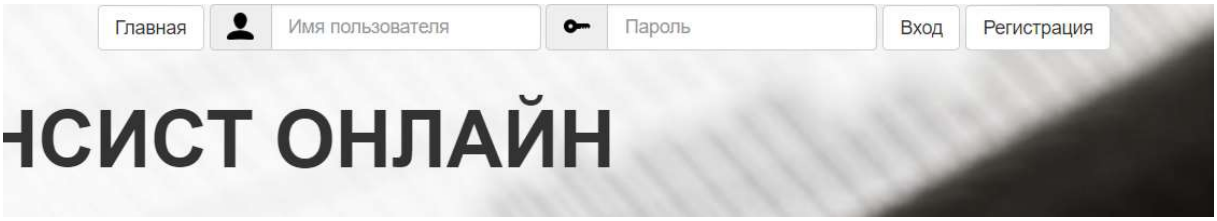


Рисунок 3.5 – Шапка сайта для неавторизованного пользователя



Рисунок 3.6 – Шапка сайта для авторизованного пользователя

Шапка сайта, как и кнопки, разработана с использованием Bootstrap. Сама шапка представляет собой форму в одну строку (класс «form-inline»). Поля для ввода логина и пароля находятся внутри группы ввода.

Разработанный дизайн в дальнейшем будет применяться на всех страницах веб-сервиса.

3.2 Реализация кредитного онлайн калькулятора

Кредитный калькулятор – программа, которая позволяет рассчитать размер выплат по кредиту, получить график предстоящих платежей, а также узнать размер переплаты. Обычно, для получения результата нужно заполнить соответствующие поля, такие как сумма займа, срок кредитования и процентная ставка.

Кредитные калькуляторы бывают двух типов: оффлайн и онлайн калькуляторы. Оффлайн калькуляторы обычно входят в состав бухгалтерских программ, их необходимо устанавливать на компьютер. Онлайн калькулятор работают как веб-приложения и размещаются в сети.

Подобных калькуляторов в сети Интернет представлено большое количество, большинство из них рассчитывают сумму ежемесячного платежа по кредиту на основе двух схем: аннуитетные или дифференцированные платежи. На разрабатываемом веб-сервисе дополнительно реализован расчет ежемесячного платежа по третьей схеме, когда погашение кредита происходит неравномерно – гибкий график погашения.

3.2.1 Верстка интерфейса калькуляторов

Все три ссылки выбора схемы погашения кредита (рисунок 3.7) будут вести на одну страницу, где в зависимости от значения переданного параметра *type_platezh* пользователь увидит форму, содержащую название калькулятора, а также поля для ввода данных по кредиту.

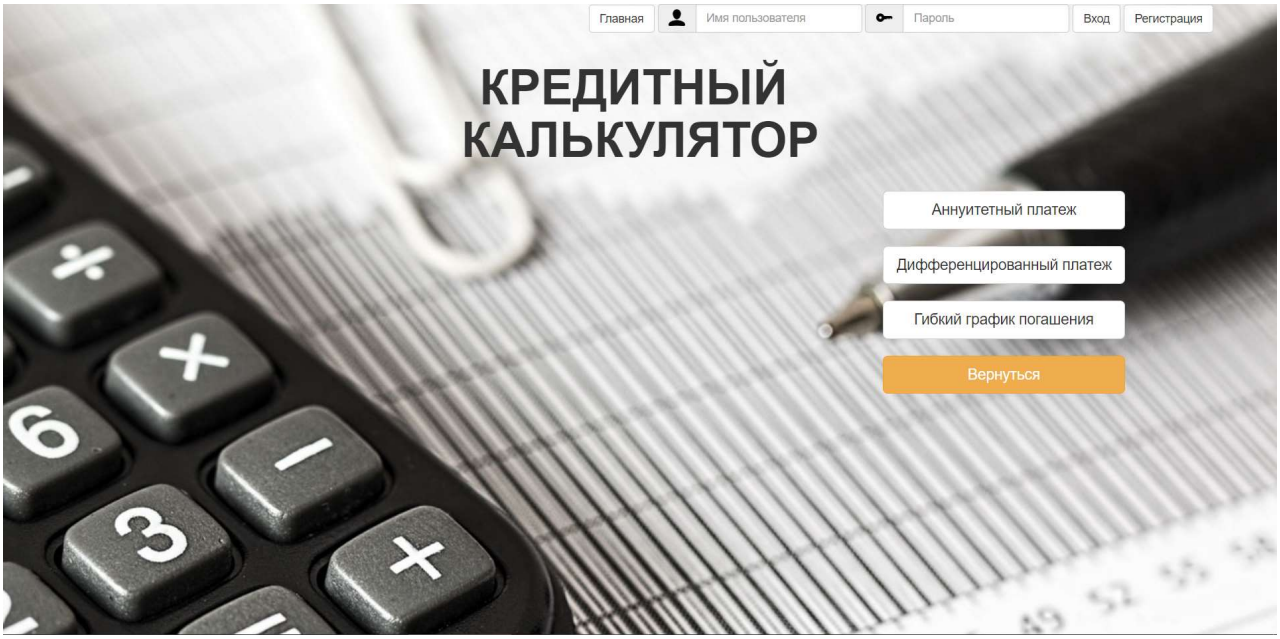


Рисунок 3.7 – Выбор схемы погашения кредита

Для кредитных калькуляторов, рассчитывающих сумму платежа на основе аннуитетных или дифференцированных платежей, достаточной информацией для проведения расчетов будет:

- дата получения кредита;
- сумма кредита;
- срок кредита в месяцах;
- процентная ставка в год.

Для расчета по гибкому графику погашения, кроме информации для предыдущих типов платежей, необходимо предусмотреть дополнительные поля, для ввода суммы гашения за каждый месяц существования кредитного договора.

Все кнопки и поля для ввода данных о кредите будут находиться в компоненте Bootstrap – Jumbotron. Он предназначен для создания блока информации на веб-сайте, на который нужно обратить внимание пользователя. Для создания Jumbotron нужно заключить содержимое в блочный элемент *div,* после чего применить к нему класс «*.jumbotron»*. Разработанная форма, на примере калькулятора, выполняющего расчёт по схеме аннуитетных платежей, представлена на рисунке 3.8.

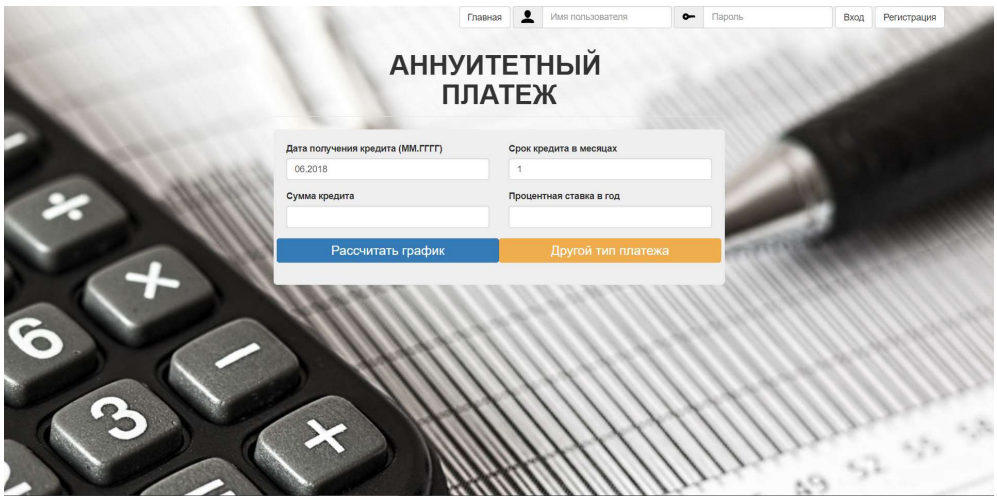


Рисунок 3.8 – Интерфейс калькулятора

Для вывода графика платежей, проверки значений, а также изменения количества полей для схемы гибких платежей используется библиотека jQuery.

3.2.2 Разработка с использованием библиотеки jQuery

jQuery – библиотека JavaScript, основной целью использования которой является взаимодействие между HTML и JavaScript. Она позволяет легко манипулировать любыми HTML-элементами документа и делать AJAX-запросы [9]. Если провести аналогию между CSS и jQuery, то можно сказать, что CSS позволяет отделить визуализацию от структуры HTML, а jQuery, в свою очередь, отделяет поведение от структуры HTML. Основные возможности jQuery [2]:

- работа с событиями;
- реализация различных визуальных эффектов;
- обращение к любым DOM-элементам и манипуляция ими;
- работа с AJAX.

Кроме того, библиотека jQuery содержит множество JavaScript-плагинов, упрощающих создание пользовательских интерфейсов.

После каждого изменения содержимого полей происходит проверка соответствия введенного значения регулярному выражению и требованию

заполнения. Если какое-то поле заполнено неверно, то около него высвечивается сообщение, с информацией о возникшей ошибке (рисунок 3.9). Ошибка исчезнет сразу после изменения значения на правильное.

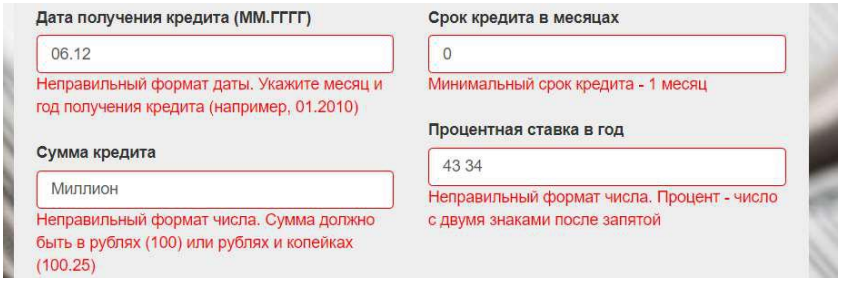


Рисунок 3.9 – Форма с неправильно заполненными полями

В настройках для проверки значения полей есть два правила, а именно: required и pattern. В первом указывается обязательность заполнения данного поля. Во втором регулярное выражение, которому должно соответствовать значение. В зависимости от того, какое из правил не выполнено, у поля отображается соответствующее сообщение из блока messages. Код настройки проверки значения одного из полей:

```
$(document).ready(function() {
    $("#frmPlatezhParam").validate({
        rules: {
            sum_kred: {
                required: true,
                pattern : "^\\d+(\\.\\d{1,2})?$"
            },
        },
        messages: {
            sum_kred: {
                required: "Укажите сумму кредита в рублях.",
                pattern : "Неправильный формат числа. Сумма должно быть в
рублях (100) или рублях и копейках (100.25)"
            },
        }
    });
```

При помощи jQuery было реализовано создание дополнительных полей для гибкой схемы погашения кредита. После загрузки веб-страницы, проверяется тип выбранной схемы погашения кредита. Если выбран гибкий платеж, то вызывается функция для создания дополнительных полей. Также функция вызывается при изменении значения полей «Дата получение кредита» и «Срок кредита в месяцах». Количество полей рассчитывается, исходя из

значения поля «Срок кредита в месяцах». Также считывается текущее значение поля «Дата получения кредита» и на его основе около каждого дополнительного поля размещается информация о том, за какой месяц сумму погашения нужно внести в это поле. Страница с созданными полями представлена на рисунке 3.10.

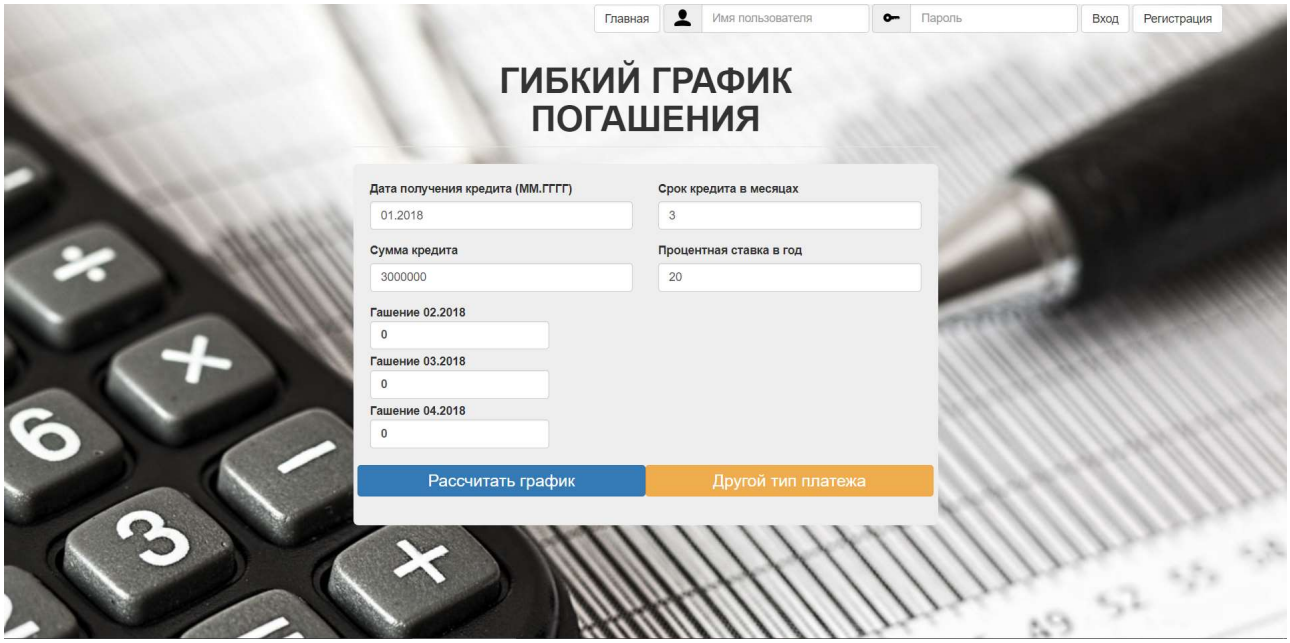


Рисунок 3.10 – Гибкий график погашения с дополнительными полями

При нажатии на кнопку «Рассчитать график», вместо вызова документа указанного в атрибуте формы – *action,* с помощью jQuery проверяются все введенные значения, чтобы не возникло ситуации, когда введенные суммы платежей по кредиту больше суммы кредита с учетом процентов, и если все верно, то посылается POST запрос к файлу, который должен выполнить все расчеты графиков. Для отправки запроса серверу используется асинхронная технология AJAX, позволяющая обращаться к серверу без обновления всей страницы [4]. Благодаря этому уменьшается время отклика и веб-сервис становится похож на настольные приложения. Самый простой пример использования этой технологии – это современные поисковые системы. Пользователь еще только заполняет поисковую строку, а сервис уже находит и предлагает возможные варианты для поиска.

Код функции, которая осуществляет запрос к php-файлу, выполняющему расчет графиков:

```
//Нажимаем на кнопку в форме
      $('#btnShowPaymentSchedule').click(function(){
            //Проверяем корректность значений в полях формы
            if (ValidFormPlatezhParam()) {
                  var data = $("#frmPlatezhParam :input").serialize();
                  $.post($("#frmPlatezhParam").attr('action'), data,
function(html_reply){
                        $('#text-popup').html(html_reply);
                        $.magnificPopup.open({
                              items: {
                                    src : "#text-popup",
                                    type: "inline"
                              }
                        });
                  }
                  )
            }
      });
```

Вызываемый файл содержит функции для расчета графиков всех видов платежей, а также сохранение графиков в pdf и xls файлы. В ответе на запрос содержится готовый HTML-код, для отображения графиков. Полученный код отображается в всплывающем окне Magnific Popup. Это jQuery-плагин всплывающих окон, позволяющий быстро создавать удобные и красивые окна. Плагин имеет гибкую настройку и поддержку адаптивного дизайна.

Результат работы этой функции представлен на рисунке 3.11.

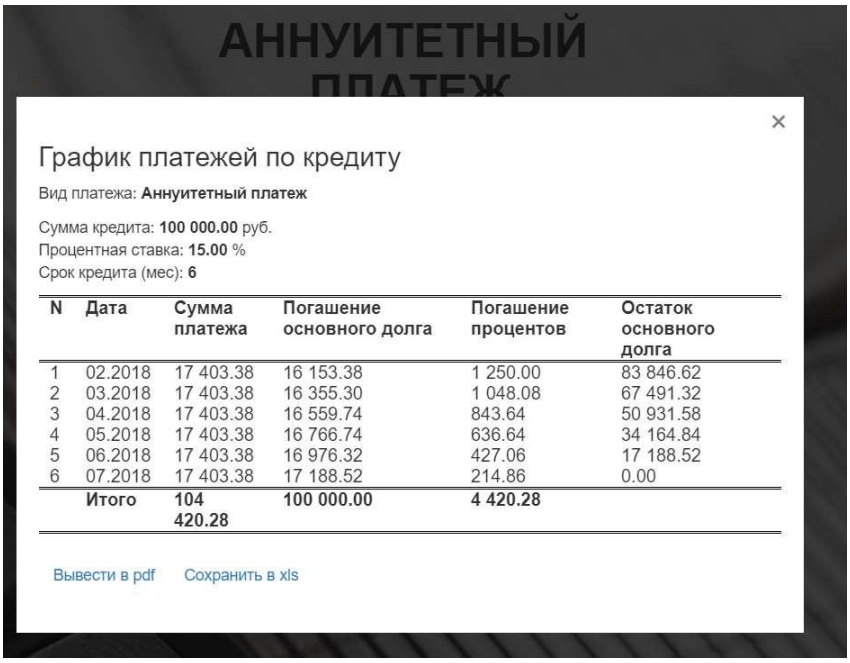


Рисунок 3.11 – График платежей по кредиту

3.2.3 Реализация алгоритмов расчёта графиков платежей.

Общий алгоритм работы серверной части при нажатии пользователем кнопки «Рассчитать график» выглядит следующим образом:

1) В переменные считываются данные полученные из POST-запроса.

2) Вызывается функция, выполняющая расчет графика платежей, который в результате возвращается массивом.

3) Вызывается функция, осуществляющая формирование HTML-кода, для вывода графика на странице. Возвращается HTML-код.

4) Отправка в ответ на запрос готового HTML-кода для вывода графика в всплывающее окно.

Как было описано ранее, в разрабатываемом калькуляторе есть поддержка трех схем платежей по кредиту: аннуитетный платеж, дифференцированный платеж и гибкий график погашения. Расчет графика платежей осуществляет функция, описанная в втором шаге алгоритма. Рассмотрим реализацию этой функции подробнее.

Первая схема погашения – аннуитетный платеж. Это ежемесячный платеж равными частями в течении всего срока действия кредитного договора [11]. Сумма ежемесячного платежа включает в себя сумму начисленных процентов и часть основного долга.

Ниже представлена формула расчета аннуитетного платежа:

$$x = S \times (P + \frac{P}{(1+P)^N - 1})$$

В этой формуле:

- x – сумма ежемесячного платежа;

- S – сумма кредита;

- P – $\frac{1}{12}$ процентной ставки;

- N – количество месяцев, которые будет выплачиваться кредит.

Первым делом рассчитываются начальные значения, такие как первый месяц оплаты кредита и месячный процент. Платежи за кредит начинаются со

следующего месяца за получением кредита. Добавление необходимого количества месяцев к дате осуществляет функция *Add_month*. Код функции *Add_month*:

```
function Add_month($time, $num=1) {
    $d=date('j',$time);  // день
    $m=date('n',$time);  // месяц
    $y=date('Y',$time);  // год
    $m+=$num;
    if ($m>12) {
        $y+=floor($m/12);
        $m=($m%12);
        // Дополнительная проверка на декабрь
        if (!$m) {
            $m=12;
            $y--;
        }
    }
    // Это последний день месяца?
    if ($d==date('t',$time)) {
        $d=31;
    }
    // Открутить дату, пока она не станет корректной
    while(true) {
        if (checkdate($m,$d,$y)){
            break;
        }
        $d--;
    }
    // Вернуть новую дату в TIMESTAMP
    return mktime(0,0,0,$m,$d,$y);
}
```

Далее осуществляется расчет месячной процентной ставки по формуле:

```
$month_proc = ($proc/100)/12;
```

Следующим шагом является расчет аннуитентного платежа, по формуле выше. Реализация этой формулы на php:

```
$platezh = $sum_kred*($month_proc + $month_proc /(pow(1+$month_proc,
$col_month)-1));
```

Далее, используя цикл, формируется график платежей для каждого месяца. Информация для каждого месяца включает в себя:

- номер платежа
- месяц платежа
- сумма платежа
- размер платежа, идущий в погашение основного долга
- размер платежа, идущий в погашение процентов

- остаток основного долга, после оплаты в этом месяце.

При аннуитентной схеме погашения сумма платежа всегда постоянная. В эту сумму входит две составляющих: погашение основного долга и погашение процентов. Часть суммы, идущая в погашение процентов, рассчитывается по формуле:

```
$platezh_proc = $ostatok_dolg*$month_proc;
```

А в погашение основного долга идет все оставшаяся сумма, получаемая вычитанием из платежа суммы, идущей в погашение процента:

```
$platezh_main_dolg = $platezh - $platezh_proc;
```

Остаток основного долга определяется вычитанием из размера долга суммы, идущей в погашение долга:

```
$ostatok_dolg = $ostatok_dolg - $platezh_main_dolg;
```

Далее все данные сохраняются в массив, который возвращается в точку вызова функции.

Вторая схема погашения – дифференцированный платеж. Этот способ погашения кредита представляет собой неравные ежемесячные платежи, которые пропорционально уменьшаются на протяжении всего срока кредитования. Банк делит весь кредит на равные части и добавляет к каждой из них проценты, исходя из срока пользования кредитом [13].

Расчет размера платежа, идущего в счет погашения основного долга, рассчитывается простым образом. Вся сумма кредита делится на количество месяцев, за которые будет погашаться кредит. Формула расчета основного платежа:

$$b = \frac{S}{N}$$

В этой формуле:

- b – основной платеж;

- S – размер кредита;

- N – количество месяцев.

Расчет размера платежа, идущего в счет погашения процентов, осуществляется по формуле:

$$p = \frac{S_n \times P}{12}$$

В этой формуле:

- p – размер платежа, идущий в погашение процентов;

- S_n – остаток задолженности на данный период;

- P – годовая ставка по кредиту.

Расчет графика погашения высчитывается аналогично графику погашения аннуитентного типа платежей, за одним исключением. Там размер выплаты в счет погашения основного долга высчитывался при помощи вычитания из суммы платежа размера платежа в счет погашения процентов. Здесь это значение уже известно и постоянно, но нужно рассчитать сам размер платежа. Размер каждого платежа будет вычисляться по формуле:

```
$platezh = $platezh_main_dolg + $platezh_proc;
```

Последняя используемая схема погашения кредита – гибкий график погашения. Этот график погашения наиболее удобен для сезонного бизнеса, так как там часто доход сильно сокращается в некоторые месяца. Гибкий график платежей разрабатывается отдельно для каждого заемщика. Для каждого месяца выплаты кредита указывается тот размер основного долга, который будет выплачен в этом месяце.

Расчет графика погашения по этой схеме практически полностью аналогичен расчету графика дифференцированных платежей. Единственное отличие, что сумма платежа в погашение основного долга нигде не высчитывается, а считывается из массива, полученного с формы через POST.

3.2.4 Сохранение графиков в pdf

На данный момент существуют 3 основных решения для генерации pdf-файлов со стороны сервера – это библиотеки PDFLib, FPDF и TCPDF. Однако

PDFLib в бесплатной версии не позволяет работать с UTF-8, что является его безусловным минусом. Аналогичная история с FPDF. Именно поэтому, для сохранения графиков платежей будет использован открытый класс TCPDF. Этот класс представляет собой полностью реализованный на PHP очень мощный механизм создания PDF-документов. Он поддерживается все современные стандарты документа, такие как: аннотации, ссылки, UTF-8, режим рендеринга текста, наборы шрифтов, интервал, пользовательский форма страниц, растягивание текста, сжатие страниц и многие другие.

Существует две версии: для PHP4 и для PHP5. Для разрабатываемого веб-сервиса подходит версия для PHP5. Установка этого класса выполняется очень просто. Сначала класс скачивается с веб-сайта для разработки и распространения открытых проектов SourceForge. Затем архив распаковывается на необходимую папку на сервере, и уже готов к подключению из любого документа.

В стандартном наборе класса отсутствуют кириллические шрифты, но это не проблема, так как класс позволяет добавить любой шрифт, поместив его в папку *fonts* внутри папки, содержащей библиотеку класса.

Для сохранения графиков в pdf реализована функция PDF. Эта функция получает график платежей, сформированный функцией, описанной в предыдущей главе, и сразу отправляет готовый pdf-документ на скачивание. Вид сформированного в проекте pdf-документа представлен на рисунке 3.12.



Финансист

Вид платежа: **Аннуитетный платеж**

Сумма кредита: **3 000 000.00** руб.
Процентная ставка: **12.00** %
Срок кредита (мес): **6**

N	Дата	Сумма платежа	Погашение основного долга	Погашение процентов	Остаток основного долга
1	07.2018	517 645.10	487 645.10	30 000.00	2 512 354.90
2	08.2018	517 645.10	492 521.55	25 123.55	2 019 833.35
3	09.2018	517 645.10	497 446.77	20 198.33	1 522 386.58
4	10.2018	517 645.10	502 421.23	15 223.87	1 019 965.35
5	11.2018	517 645.10	507 445.45	10 199.65	512 519.90
6	12.2018	517 645.10	512 519.90	5 125.20	0.00
	Итого:	3 105 870.60	3 000 000.00	105 870.60	

Рисунок 3.12 – Готовый график в pdf-документе

3.2.5 Сохранение графика платежей в xls

Для сохранения графиков платежей в xls будет использована библиотека PHPExcel. Она предназначена для обработки данных формата MS Excel. С ее помощью можно записывать информацию в файлы, выполнять операции с формулами и стилями, форматировать содержимое документов.

Установка библиотеки осуществляется просто. Необходимо скачать библиотеку с официального сайта, а потом распаковать ее в необходимую папку на сервере. Для сохранения графика платежей в Excel реализована функция XLS(). В результате пользователь получает файл формата xls, содержащий график платежей. Готовый файл представлен на рисунке 3.13.

	A	B	C	D	E	F
4	Срок кредита: 12 мес.					
5	N	Дата	Сумма платежа	Погашение основного долга	Погашение процентов	Остаток основного долга
6	1	07.2018	799639,1	709639,1	90000	8290360,9
7	2	08.2018	799639,1	716735,49	82903,61	7573625,41
8	3	09.2018	799639,1	723902,85	75736,25	6849722,56
9	4	10.2018	799639,1	731141,87	68497,23	6118580,69
10	5	11.2018	799639,1	738453,29	61185,81	5380127,4
11	6	12.2018	799639,1	745837,83	53801,27	4634289,57
12	7	01.2019	799639,1	753296,2	46342,9	3880993,37
13	8	02.2019	799639,1	760829,17	38809,93	3120164,2
14	9	03.2019	799639,1	768437,46	31201,64	2351726,74
15	10	04.2019	799639,1	776121,83	23517,27	1575604,91
16	11	05.2019	799639,1	783883,05	15756,05	791721,86
17	12	06.2019	799639,08	791721,86	7917,22	0
18		Итого:	9595669,18	9000000	595669,18	
19						
20						
21						

График платежей

Рисунок 3.13 – График платежей в готовом документе Excel

3.3 Реализация базы данных

При разработке проекта большую роль играет грамотная реализация структуры базы данных. В разработанной в разделе 1.5.3 логической модели данных были учтены основные составляющие проекта, определены необходимое количество таблиц, содержащиеся в них данные и существующие связи. Теперь необходимо создать базу данных, соответствующую разработанной схеме. Для этого использовался стандартный инструмент администрирования СУБД MySQL – phpMyAdmin (рисунке 3.14).

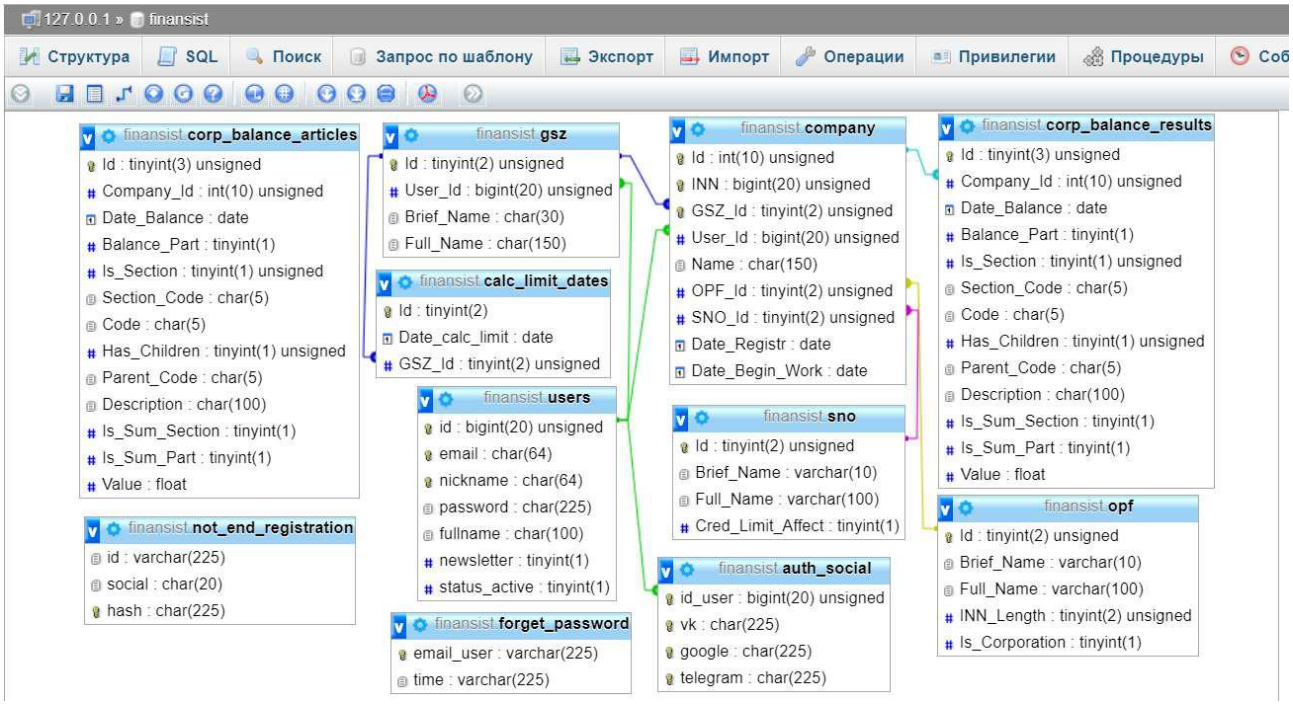


Рисунок 3.14 – База данных проекта в «Дизайнере» phpMyAdmin

3.4 Реализация авторизации и регистрации на сервисе

Как описывалось ранее, для расчета максимальной суммы кредита необходимо реализовать авторизацию и регистрацию на сервисе. Все необходимые таблицы в базе данных уже были созданы в предыдущем пункте, поэтому осталось создать интерфейс и серверные функции.

Необходимо разграничить доступ к различным страницам веб-сервиса для пользователей в зависимости от того, авторизован пользователь, или нет. Так, например, авторизованный пользователь не должен попасть на страницы авторизации и регистрации, а пользователь, не прошедший авторизацию, не должен попасть на страницу настроек профиля или списка компаний даже по ссылкам. Поэтому доступ к основным страницам раздела авторизации и регистрации будет осуществляться через скрипты, которые выполняют роль маршрутизации по страницам.

Так, например, скрипт login.php проверяет, авторизован ли пользователь. Если пользователь авторизован, то происходит переадресация на главную страницу.

```
if (is_current_user()) {
      redirect('./');
}
```

В случае если пользователь еще не авторизовался, то проверяется начал ли пользователь процесс авторизации. В случае, если пользователь только хочет авторизоваться, ему выводится пустая форма для авторизации (рисунок 3.15). В ситуациях, когда страница вызвана путем переадресации от социальной сети, о чем сообщает значение атрибута GET-массива – *log,* либо нажатием на кнопку «Войти», после ввода данных, если в POST-массиве существует атрибут *login,* то полученные данные обрабатываются, проверяются и выносится решение – авторизовать пользователя, либо нет. Если решение положительное, то данные о пользователе сохраняются в сессию и происходит переадресация на главную страницу, а если отрицательное, то выводится страница с ошибками (рисунок 3.16), либо, в случае, когда пользователь попытался войти через социальную сеть, но данные о нем не найдены в базе данных, то форма регистрации.

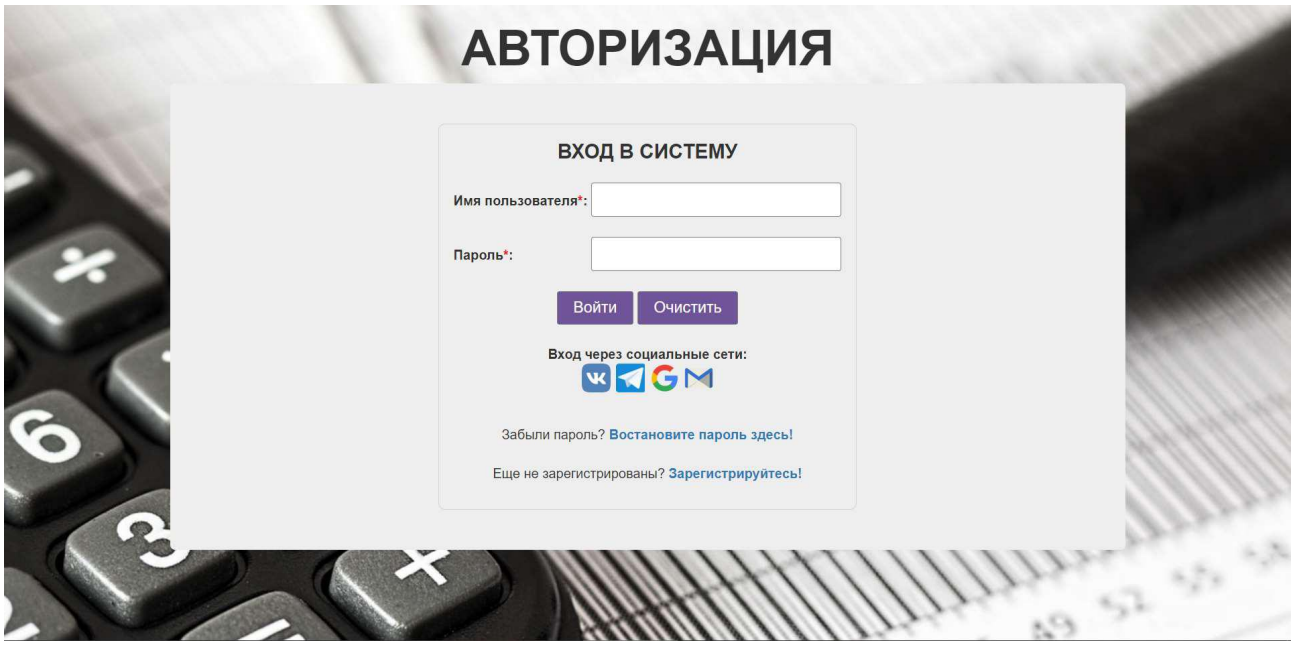


Рисунок 3.15 – Форма «Вход в систему»

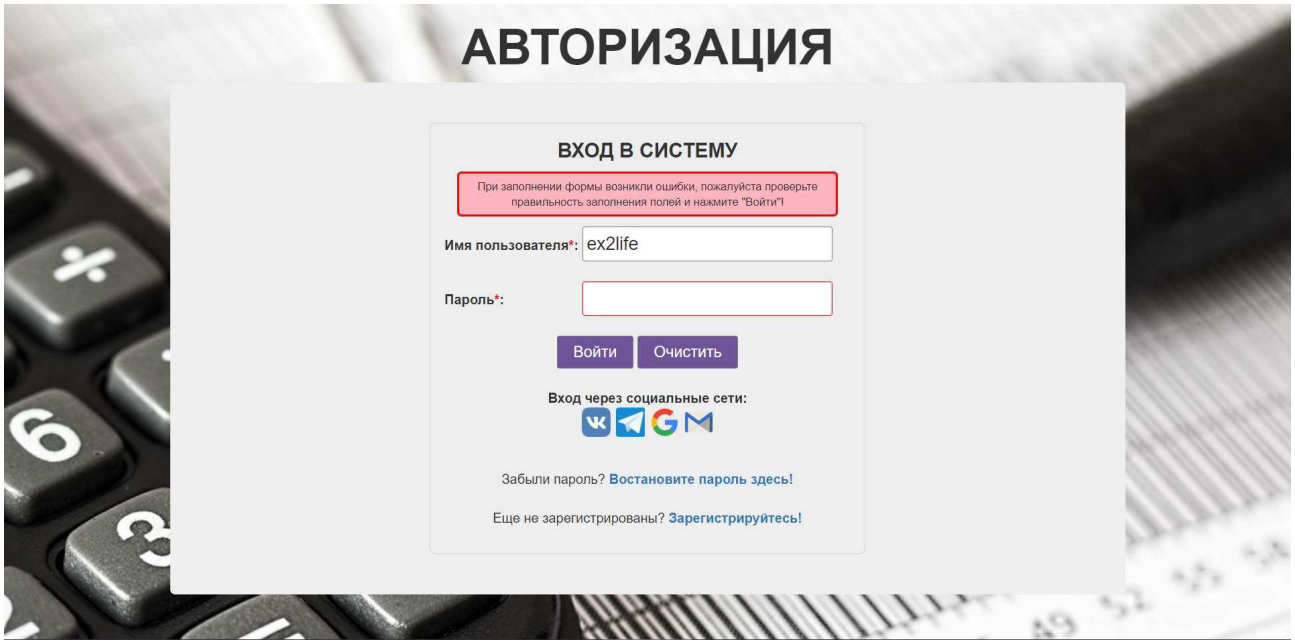


Рисунок 3.16 – Форма «Вход в систему» с ошибками

Для регистрации пользователь должен с любой страницы сервиса нажать кнопку в шапке «Регистрация», в результате чего он попадет на соответствующую форму (рисунок 3.17). Страница регистрации представляет собой панель, где необходимо ввести имя пользователя, адрес электронной почты, пароль, подтверждение пароля, ФИО, после чего нажать кнопку «Зарегистрироваться».

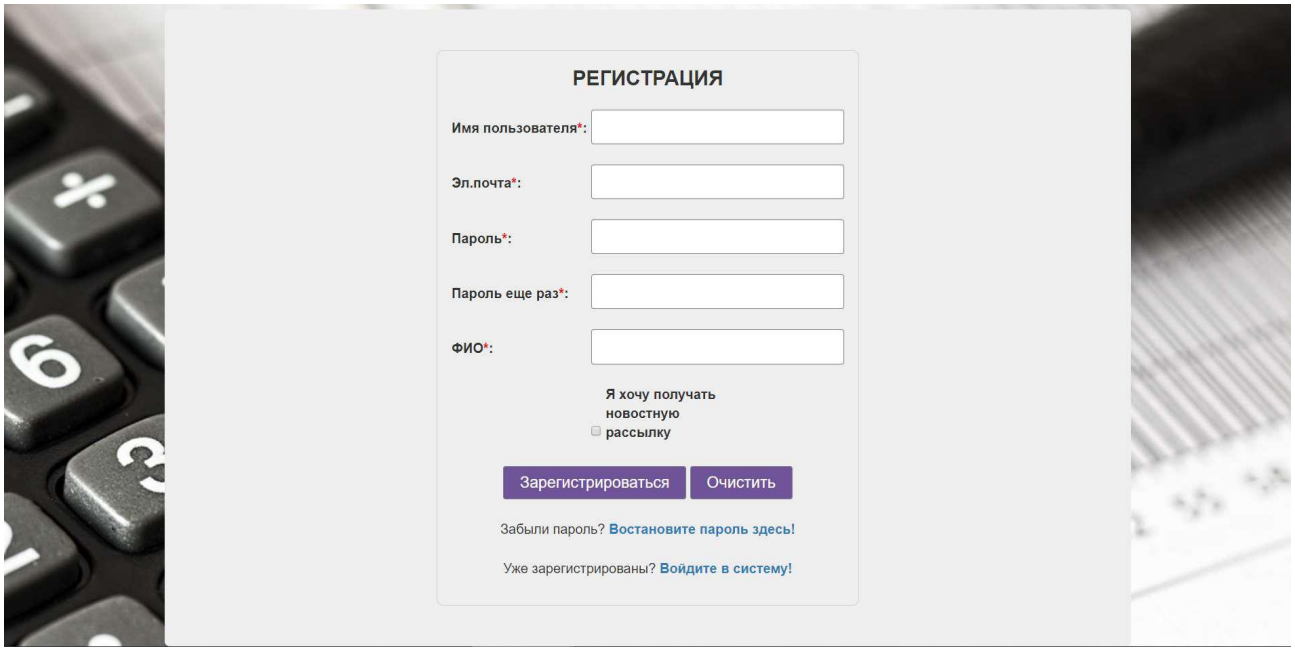


Рисунок 3.17 – Форма регистрации

Первым этапом в процессе регистрации пользователя является проверка введенных им данных. Для этого реализованы 3 функции:

- read_email() – функция для проверки корректности адреса электронной почты

- read_bool() – функция проверки корректности логических значений

- read_string() – универсальная функция для проверки корректности любых значений строковых типов данных. В качестве атрибутов принимает требования к проверяемому значению, такие как обязательно ли должно быть заполнено поле, минимальная и максимальная допустимая длина для значения.

Код функции read_string():

```
function read_string($form, $field, &$obj, &$errors, $min, $max,
$is_required, $default=null, $trim=true)
{
	$obj[$field] = $default;
	if (!isset($form[$field])) {
		return $is_required ? add_error($errors, $field, 'required') :
true;
	}
	$value = $trim ? trim($form[$field]) : $form[$field];
	if ($value == '' && $is_required)
		return add_error($errors, $field, 'required');
	if (strlen($value) < $min)
		return add_error($errors, $field, 'too-short');
	if (strlen($value) > $max)
		return add_error($errors, $field, 'too-long');
	$obj[$field] = $value;
	return true;
}
```

Код функции read_email():

```
function read_email($form, $field, &$obj, &$errors, $min, $max,
$is_required, $default=null){
	$obj[$field] = $default;
	if (!isset($form[$field])) {
		return $is_required ? add_error($errors, $field, 'required') :
true; }
	$value = trim($form[$field]);
	if (strlen($value) < $min)
		return add_error($errors, $field, 'too-short');
	if (strlen($value) > $max)
		return add_error($errors, $field, 'too-long');
	// проверяем, что в строке задан адрес электронной почты
	if (!filter_var($value, FILTER_VALIDATE_EMAIL))
		return add_error($errors, $field, 'invalid');
	$obj[$field] = $value;
	return true;
}
```

Код функции read_bool():

```
function read_bool($form, $field, &$obj, &$errors, $true, $is_required,
$default=null)
{
      $obj[$field] = $default;
      if (!isset($form[$field])) {
            return $is_required ? add_error($errors, $field, 'required') :
true;
      }
      $value = trim($form[$field]);
      $obj[$field] = $value === $true;
      return true;
}
```

Если в результате проверки оказалось, что какое-то из значений заполнено неверно, либо в результате дальнейшей проверки пароль не совпал с подтверждением пароля, то при помощи функции add_error() в массив ошибок $errors добавляется название ошибки и информация о ней.

Код функции add_error ():

```
function add_error(&$errors, $field, $description)
{
      $errors['fields'][] = $field;
      $errors['messages'][$field] = "@$field-$description";
      return false;
}
```

Предусмотрена проверка адреса электронной почты и псевдонима пользователя на занятость. Если в результате проверки будет обнаружено, что пользователь ввел данные, которые уже присутствуют в системе, то в массив ошибок будет также добавлена информация об этом.

После всех проверок, с помощью функции has_error() проверяется, содержит ли массив $errors ошибки. Если в результате хотя бы одной проверки были добавлены в массив ошибки, то функция регистрации завершается и пользователю выводится страница регистрации с теми данными, которые он заполнял, и подсвеченными полями, где он допустил ошибку. Для некоторых полей предусмотрен вывод информации о ошибке. Пример страницы с неправильно заполненными полями приведен на рисунке 3.18.

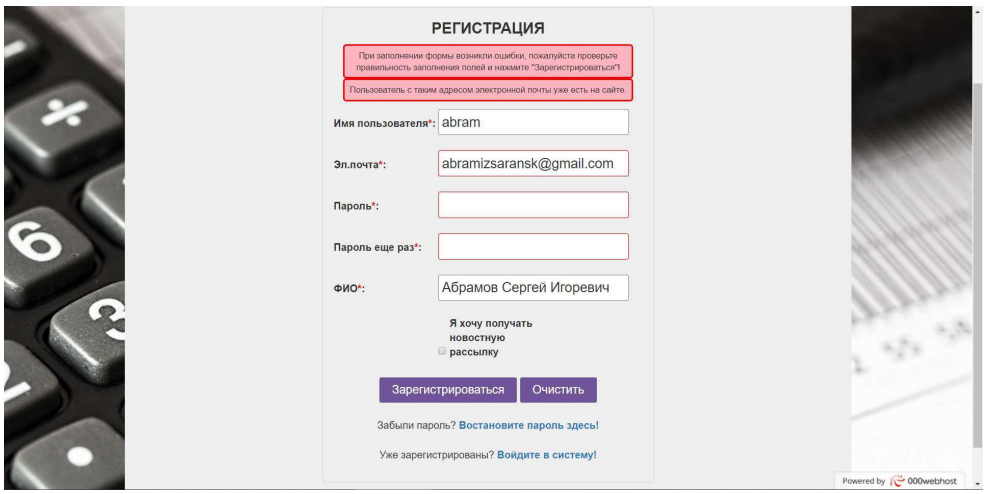


Рисунок 3.18 – Форма регистрации с возникшими ошибками

Если массив с ошибками оказался пуст, то наступает следующий этап регистрации – защита пароля пользователя.

Существует три способа хранения пароля:

- хранение пароля в том виде, в котором его ввел пользователь;

- хранение пароля в зашифрованном виде, используя алгоритмы шифрования;

- добавление к паролю несколько случайных символов, называемых «солью», и хранение полученного результата в зашифрованном виде.

Первый способ наиболее неблагоприятный для использования, так как при взломе базы данных злоумышленник получает все пароли в открытом виде, что может серьезно навредить пользователям.

Второй способ более удачный. При взломе базы данных злоумышленник получит только зашифрованные пароли и не сможет их расшифровать, так как шифрование необратимое. Однако, путем перебора различных паролей злоумышленник может сравнивать украденный хэш с полученным хэшем с помощью перебора, и довольно велика вероятность подобрать нужный пароль. Кроме того, существуют сервисы, в которых хранятся миллиарды хешей для самых популярных паролей. Например, подобный сервис подобрал пароль «Saransk» по его md5-хешу менее чем за секунду.

Третий способ самый безопасный. Для каждого пользователя генерируется уникальный набор символов, так называемая «соль», которая добавляется к паролю пользователя. Готовый результат шифруется и сохраняется в базу данных. Именно этот и способ будет использован в разрабатываемом сервисе.

Создать хеш пароля можно различными способами. Самые популярные из них – md5 и sha1. Оба алгоритма до сих пор не взломаны и существует только вероятность подбора пароля путем брутфорса. Однако в данном случае возникает вопрос об использовании «соли». Для того, чтобы в дальнейшем сравнить введенный пароль с хешем в базе данных, необходимо также хранить где-то случайно сгенерированную соль в открытом виде. Чаще всего для этого используют базу данных, однако возникает вероятность утечки «солей», что сведет на ноль безопасность хранимых паролей, достигнутую с их использованием. К тому же, случайная «соль» тоже может быть подобрана. Компьютер – детерминированная машина, т.е. любой выполняемый алгоритм, получив несколько раз одни и те же данные, всегда предоставит один и тот же вариант. Поэтому все случайные данные сгенерированные компьютером называются псевдослучайными, и существует вероятность воссоздания набора исходных состояний, когда результатом генерации будет выступать искомая «соль».

Именно поэтому, для шифрования паролей в разрабатываемом сервисе, принято решение использовать функцию password_hash(). Функция поддерживает три алгоритма шифрования: bcrypt, CRYPT_BLOWFISH и Argon2 [15]. Самый безопасный алгоритм хранения из представленных, это используемый стандартно bcrypt. Важной особенностью использования этого алгоритма является то, что алгоритм может меняться на более сильный, когда таковой добавляется в PHP. При этом, все хешы паролей, созданные более старыми версиями алгоритма, можно без проблем проверить и новой версией, так как в хеше содержится информация о используемой для шифрования версии.

Алгоритм позволяет задать свою «соль» при хешировании, однако наилучшим и наиболее безопасным выбором является предоставить создание «соли» самому алгоритму, это позволяет избежать проблем с ее хранением.

Проверка пароля осуществляется при помощи функции password_verify(), принимающей два аргумента: пароль и хеш для проверки.

В случае обновления алгоритма на более сильный, существует специальная функция password_needs_rehash(), позволяющая проверить необходимость обновления хэша.

После обеспечения безопасности для пароля следующий этап регистрации – добавление информации о пользователе в базу данных. Это осуществляется при помощи функции db_user_insert().

Код функции:

```
function db_user_insert($dbh, $user)
{     $query = 'INSERT INTO
users(nickname,email,password,fullname,newsletter,status_active)
VALUES(?,?,?,?,?,?)';
      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);
      $user['status_active']=0;
      mysqli_stmt_bind_param($stmt, 'ssssii',
            $user['nickname'], $user['email'], $user['password'],
            $user['fullname'], $user['newsletter'],
$user['status_active']);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем идентификатор вставленной записи
      $user['id'] = mysqli_insert_id($dbh);

      // освобождаем ресурсы, связанные с хранением результата и запроса
      mysqli_stmt_close($stmt);

      return $user;
}
```

Для работы с базой данных используется драйвер PHP MySQLi, который позволяет с помощью предварительно объявляемых операторов полностью избежать SQL-иньекций. Все разработанные функции, работающие с базой данных, действуют по одному сценарию:

1) В переменную $query записывается SQL-выражение с метками параметров.

2) При помощи функции mysqli_prepare() SQL-выражение готовится к выполнению.

3) С помощью функции mysqli_stmt_bind_param() происходит привязка необходимых переменных к меткам параметров в SQL-выражении.

4) Выполнение подготовленного запроса функцией mysqli_stmt_execute().

5) Проверка того, что запрос выполнен успешно.

Следующий этап работы этих функций зависит от их назначения. Если функция извлекает данные из базы данных, то используя функцию mysqli_stmt_get_result() нужно получить результирующий набор строк от результата запроса. После чего с помощью функции mysqli_fetch_assoc() извлечь необходимую строку, либо циклом набор строк, в виде ассоциативного массива.

Последний этап работы этих функций – освобождение ресурсов, связанных с хранением запроса и результата при помощи функций mysqli_free_result() и mysqli_stmt_close().

После добавления информации о пользователе в базу данных, ему присваивается статус неподтвержденного адреса электронной почты. Для подтверждения адреса пользователю необходимо перейти в почтовый ящик почты, которую он указал, и перейти по ссылке для подтверждения. Для отправки письма пользователю реализована функция send_confirm_message(). В качестве атрибутов функция включает массив данных о пользователе, исходя из которых формирует заголовки для письма.

Код функции, осуществляющей отправку письма для подтверждения регистрации пользователю:

```
function send_confirm_message($user)
{
$subject="Подтверждение регистрации на сайте Финансист";
$headers = "To: name <".$to.">\r\n";
$headers = "From: Финансист\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: text/html; charset=utf-8";
```

```
$user['link']=$_SERVER['HTTP_HOST']."/registration/verification.php?mail=
".$user['email']."&hash=".md5(SALT.$user['id']);
$message = gener_email_html($user);
// Отправляем
mail($user['email'], $subject, $message,$headers);
redirect('thankyou_regist.php');
}
```

Стоит обратить внимание на атрибут ссылки hash. Он используется для того, что подтвердить, что пользователь действительно использует ссылку, сгенерированную сервисом, а не пытается воссоздать ссылку сам. Значение переменной SALT и идентификатор пользователя не известны пользователям, поэтому их md5-хэш служит достаточным способом проверки целостности ссылки.

Функция gener_email_html() формирует html-код письма, которое будет отображено на почте у пользователя. Пример письма, присланного пользователю, представлен на рисунке 3.19.

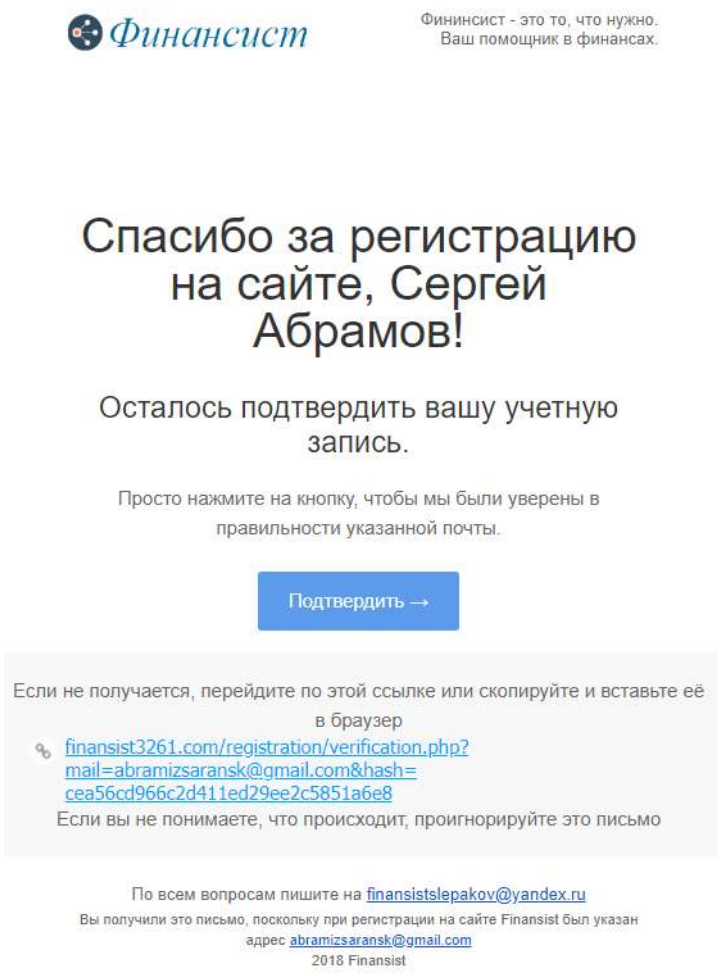


Рисунок 3.19 – Письмо для подтверждения адреса электронной почты

Ссылка из письма приводит пользователя в скрипт verification.php, где при помощи вызова функции good_email() осуществляется проверка полученных данных в GET-массиве. Если данные верны, то обновляется статус пользователя в базе данных, и пользователю выводится страница «с благодарностью» за подтверждение регистрации.

Код функции good_email():

```
function good_email($dbh, $array_get){
      $id_email=db_user_id_find_by_email($dbh, $array_get['mail']);
      if ($array_get['hash']==md5(SALT.$id_email)) {
            $status=db_user_status_update($dbh, $id_email);
            return true;
      } else {
            return false;
      }
}
```

Процесс авторизации пользователя совершает функция login_user(), в которой проверяется корректность введенных данных. Если логин и пароль заполнены верно, то совершается поиск пользователя с указанным логином (псевдоним или адрес электронной почты) по базе данных, и выгрузка строки с информацией о нем. Если такой пользователь найден в базе данных, а учетная запись подтверждена, то используя описанную ранее функцию password_verify() проверяется соответствие пароля хранимому хэшу. Если все условия соблюдены, то в сессии запоминается авторизованный пользователь.

Для работы с сессией реализованы 4 функции:

- is_current_user() – проверяет, имеется ли в сессии информация о вошедшем пользователе.

- get_current_user_id() – возвращает идентификатор пользователь, вошедшего в систему

- store_current_user_id($id) – сохраняет идентификатор в сессию

- reset_current_user_id() – сбрасывает идентификатор пользователя в сессии.

Код функций для работы с сессией:

```
function is_current_user(){
      return isset($_SESSION['user_id']);
}
```

```
function get_current_user_id(){
      return $_SESSION['user_id'];
}
function store_current_user_id($id){
      $_SESSION['user_id'] = $id;
}
function reset_current_user_id()
{     unset($_SESSION['user_id']);
}
```

Последний шаг в авторизации пользователя – проверка необходимости обновления хэша пользователя. Если функция password_needs_rehash() сообщает, что необходимо обновить хэш, то создается новый хэш из введенного пароля, и информация о пользователе обновляется в базе данных.

Полный код функции, осуществляющей процесс авторизации пользователя:

```
function login_user($dbh, &$user, &$errors)
{
      $user = array();
      $errors = empty_errors();
      // проверяем и считываем строки из запроса
      read_string($_POST, 'username', $user, $errors, 2, 64, true);
      read_string($_POST, 'password', $user, $errors, 6, 20, true);

      if (has_errors($errors))
            return false;

      // форма передана правильно, ищем пользователя и проверяем пароль
      $db_user = db_user_find_by_login($dbh, $user['username']);
      // смотрим, есть ли такой пользователь и правильно ли передан
пароль
      if ($db_user == null || !password_verify($user['password'],
$db_user['password']))
            return add_error($errors, 'password', 'invalid');
      if ($db_user['status_active']=='0')   {
            return add_error($errors, 'status_active', 'no_active');
      }
      // пользователь ввел правильные имя и пароль, запоминаем его в
сессии
      store_current_user_id($db_user['id']);
      // проверяем, нужно ли обновить хэш пароля
      if (password_needs_rehash($db_user['password'], PASSWORD_DEFAULT))
      {
     // Хэш пароля нужно обновить, в связи с изменением алгоритма
     $db_user['password'] = password_hash($user['password'],
PASSWORD_DEFAULT);
      // обновим хэш в базе данных
     $db_user = db_user_update($dbh, $db_user);
     // don't forget to store the new hash!
     }
      return true;
}
```

Также была разработана страница «Настройки профиля», где авторизованный пользователь может изменить псевдоним на любой другой свободный, исправить свое ФИО, изменить статус согласия на новостную рассылку. Также в настройках пользователь может сменить пароль. Страница «Настройки профиля» представлена на рисунке 3.21.

Для навигации по возможностям настройки профиля были использованы Bootstrap-вкладки. Это компонент, предназначенный для отображения большого количества информации на небольшой странице. Контент размещается на небольших панелях, переключение между которыми осуществляется путем переключения этих панелей с помощью вкладок.

Вкладки описываются в виде списка с классами *nav u nav-tabs,* каждый пункт которого представляет собой ссылку [14].

Блок кода, описывающий вкладки на странице «Настройки профиля»:

```
<ul class="nav nav-tabs">
     <li class="active"><a data-toggle="tab" href="#setting">Основные
настройки</a></li>
     <li><a data-toggle="tab" href="#password">Изменение пароля</a></li>
     <li class="dropdown">
<a class="dropdown-toggle" data-toggle="dropdown" href="#">Подключение
социальных сетей
          <span class="caret"></span></a>
          <ul class="dropdown-menu">
               <li ><a data-toggle="tab" href="#vk">VK</a></li>
               <li id="telegram_button"  <?php if
($_COOKIE["Telegram_blocked"]=='yes'):?>hidden<?php endif; ?>><a data-
toggle="tab" href="#telegram">Telegram</a></li>
               <li ><a data-toggle="tab" href="#google">Google</a></li>
          </ul>
     </li>
</ul>
```

Вкладка «Подключение социальных сетей» представляет собой выпадающий список, состоящий из социальных сетей, которые можно подключить для авторизации (рисунок 3.20).

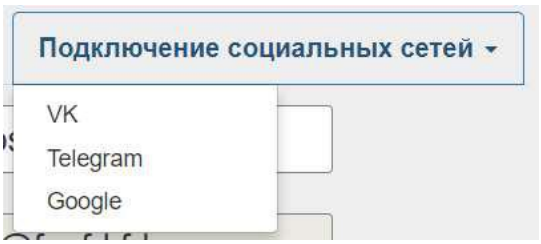


Рисунок 3.20 – Список социальных сетей для подключения авторизации

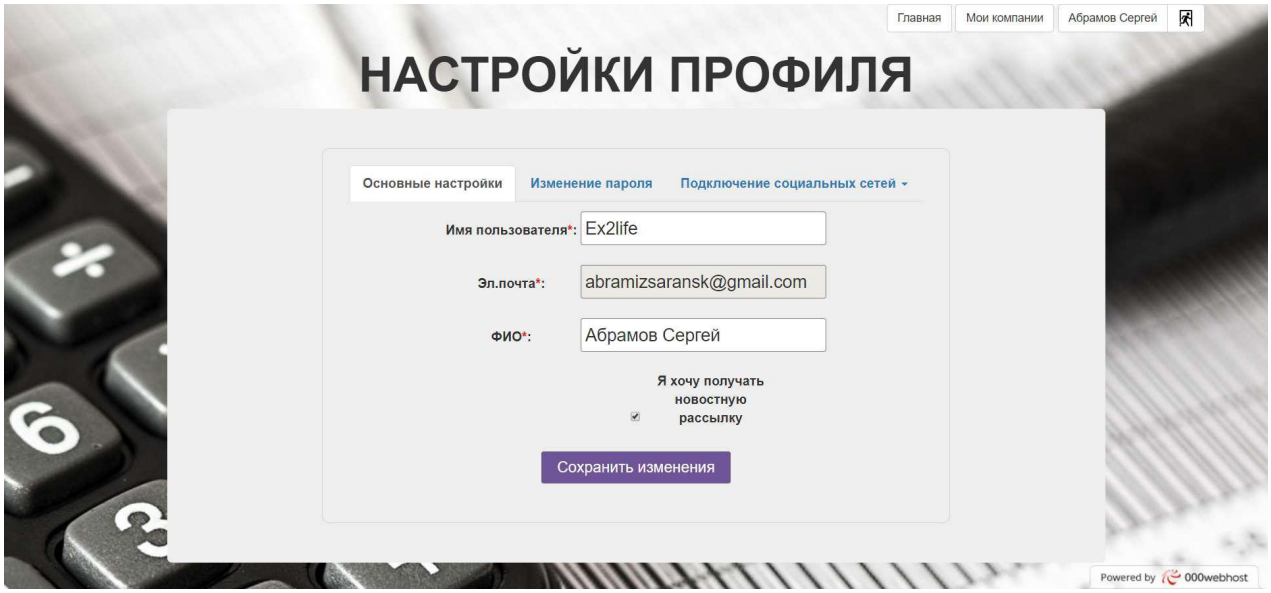


Рисунок 3.21 – Страница «Настройки профиля»

Для удобства использования сервиса реализована авторизация через три распространенные социальные сети: Вконтакте, Google и Telegram.

3.4.1 Авторизация через Вконтакте

Для авторизации через социальную сеть Вконтакте в разрабатываемом проекте используется OAuth 2.0. Это открытый протокол авторизации, позволяющий получить разработчику доступ к данным пользователя из социальной сети. При использовании этого протокола пользователь не предоставляет разработчику логин и пароль для доступа к данным социальной сети.

Данный протокол работает следующим образом:

1) С разрабатываемого веб-сервиса пользователь перенаправляется на определенный адрес в социальной сети.

2) Пользователь проходит самостоятельно авторизацию в социальной сети.

3) После авторизации социальная сеть перенаправляет пользователя обратно на веб-сервис, при этом передавая всю необходимую информацию о пользователе.

Для создания механизм авторизации через Вконтакте, разработчик должен иметь аккаунт в этой социальной сети. На первом этапе нужно создать в социальной сети приложение, так как доступ к получению данных получает именно приложение. Необходимо ввести название приложения, выбрать пункт «веб-сайт», и заполнить поля «Адрес сайта» и «Базовый домен». Процесс создание приложения приведен на рисунке 3.22. После нажатия кнопки «Подключить сайт» необходимо подтвердить операцию с помощью SMS или авторизованного мобильного приложения, после чего приложение будет создано.

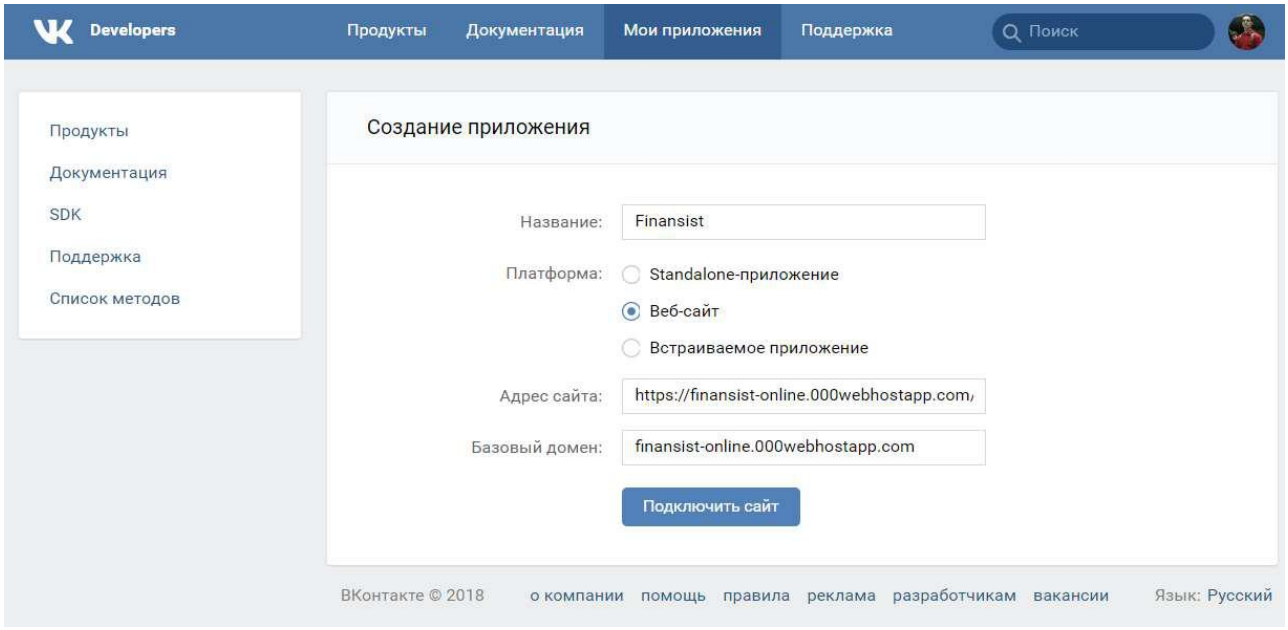


Рисунок 3.22 – Создание приложения для авторизации

После создания приложения, откроется окно для заполнения информации о приложении. В связи с тем, что приложения создается для авторизации, эта информация при разработке не является обязательной. Необходимо перейти в вкладку «Настройки».

На этой вкладке владельцу приложения доступны ID приложения, защищенный ключ, а также сервисный ключ доступа. Это секретная информация, которая не должна попадать в открытый доступ. Если есть подозрения о компрометации ключей, их необходимо обновить именно в этой вкладке, иначе они могут быть использованы злоумышленниками.

Для виджета OAuth 2.0 нужно заполнить всю необходимую информацию для Open API. В поле «Базовый домен» необходимо указать все домены, на которых будет работать разрабатываемый сервис, а также заполнить все доверенные URL для переадресации. При каждой загрузке виджет авторизации проверяет, на каком домене он запущен, а также на какой адрес необходимо после авторизации перенаправить пользователя. Если эти данные не соответствуют настройкам, указанным в этой вкладке, виджет не будет загружен.

Пример настроек для разрабатываемого веб сервиса представлен на рисунке 3.23.

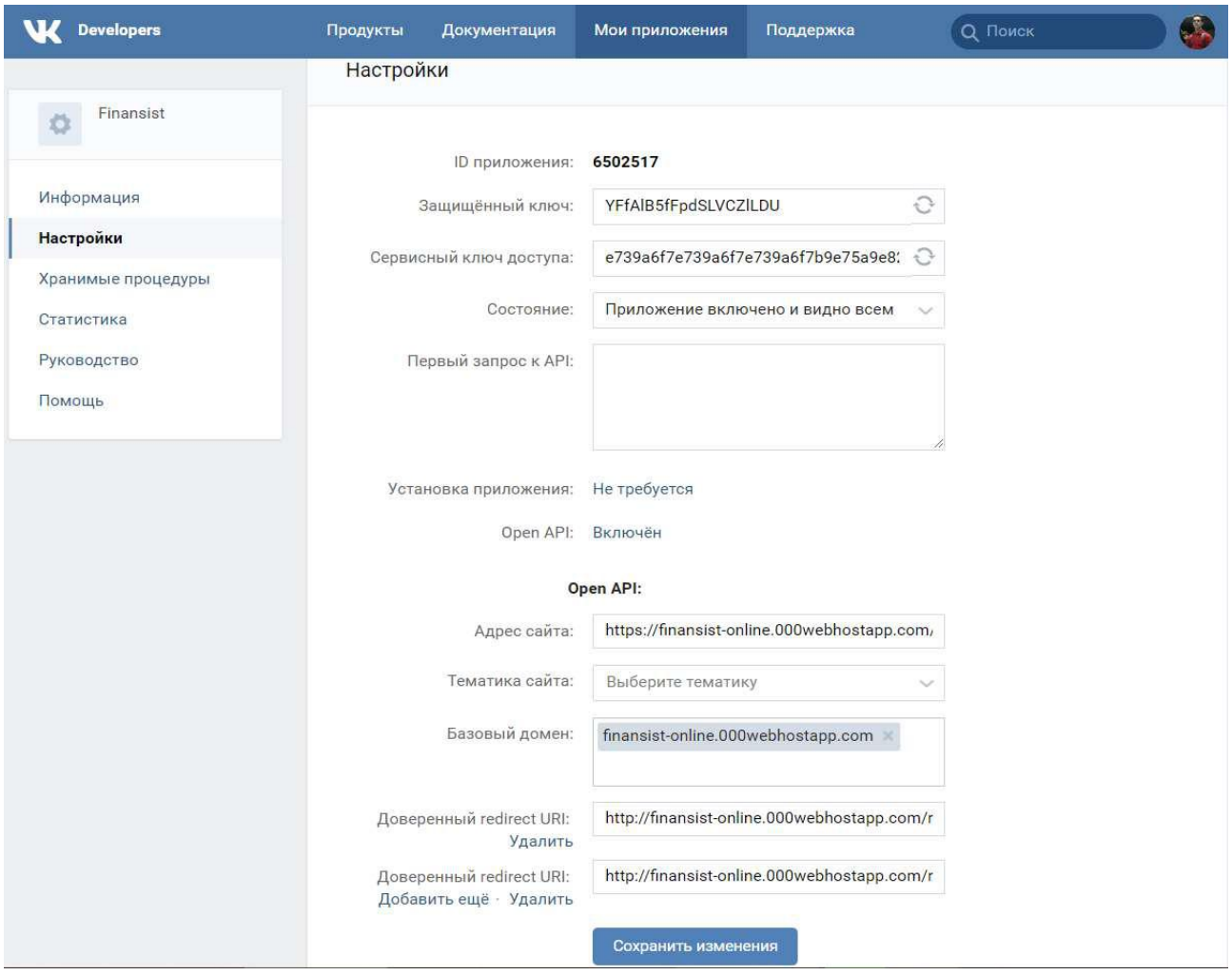


Рисунок 3.23 – Настройки приложения для авторизации

На этом настройки авторизации со стороны социальной сети закончены и можно перейти к разработке механизма авторизации.

В коде страницы, на которой выводится виджет авторизации, необходимо подключить javascript библиотку Open API от Вконтакте, а также указать идентификатор созданного раннее приложения для авторизации. Эти настройки общие для всех страниц, где будет выводиться виджет авторизации, поэтому для удобства администрирования будут вынесены в отдельный файл social_sites_config.php и подключены при возникновении потребности в этом.

Необходимый код для отобажения виджета авторизации Вконтакте:

```
<script type="text/javascript"
src="//vk.com/js/api/openapi.js?152"></script>
<script type="text/javascript">
  VK.init({apiId: 6502517});
</script>
```

Далее на каждой странице, использующей виджет авторизации, необходимо задать его настройки. В настройках указывается тип виджета, его ширина, а также адрес для переадресации пользователя после авторизации.

Пример настроек виджета для страницы «Авторизация»:

```
VK.Widgets.Auth("vk_auth",
{"width":300,"authUrl":"/registration/login.php?log=vk&"});
```

Место для вывода виджета авторизации обозначается блоком div с идентификатором vk_auth. Виджет авторизации Вконтакте представлен на рисунке 3.24.

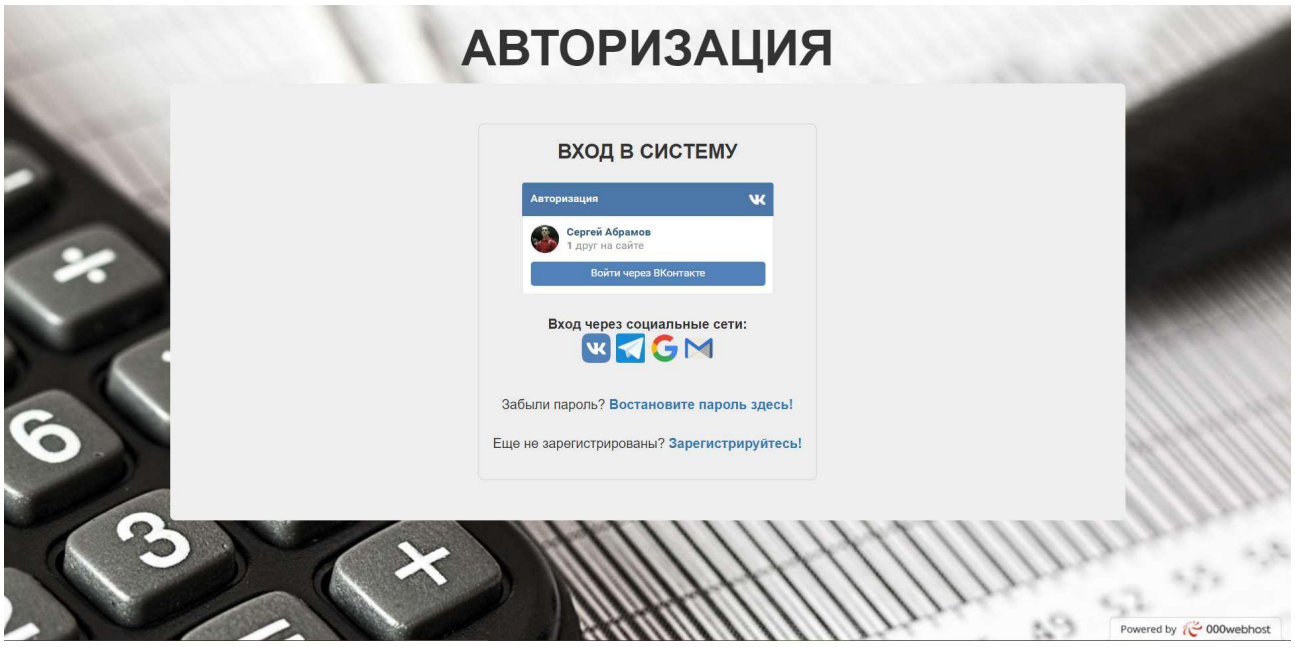


Рисунок 3.24 – Виджет авторизации ВКонтакте

Удобство использования виджета заключается в том, что пользователь сразу видит, какая учетная запись социальной сети сейчас будет использована. Нажав на кнопку «Войти через ВКонтакте», в случае первой авторизации пользователь перенаправляется в социальную сеть, где дает свое согласие на предоставление доступа к общей информации. Если согласие получено, то происходит перенаправления пользователя на адрес, указанный в настройках виджета. Все предоставленные данные о пользователе содержатся в GET-массиве. Для проверки, что данные действительно предоставлены социальной сетью, необходимо сравнить полученный параметр hash с md5-хешем от (идентификатор приложения+идентификатор пользователя+защищенный ключ). Если полученный и вычисленный хеш совпадают, то данные верны, и можно приступить к авторизации.

Полученный от социальной сети идентификатор пользователя ищется в таблице auth_social. Если информация найдена, то информация о пользователе сохраняется в сессию и пользователь считается авторизованным. Если такой информации нет в базе данных, то пользователь попадает на страницу регистрации, а идентификатор сохраняется в таблицу not_end_registration. Все данные полученные от социальной сети уже представлены в форме регистрации, что сильно ее ускоряет, и пользователю необходимо заполнить только недостающие поля.

3.4.2 Авторизация через Telegram

В начале 2018 года Telegram представил свой виджет авторизации, с помощью которого можно провести авторизацию на веб-сервисе. Когда пользователь впервые авторизуется на сайте с помощью этого виджета, Telegram запрашивает его номер телефона и отправляет ему подтверждения через приложение для авторизации браузера. Все следующие авторизации будут совершаться пользователем в несколько кликов.

Как и в случае с авторизацией через ВКонтакте, разработчик должен также иметь аккаунт в Telegram. И если там доступ к данным получает приложение, то здесь роль такого приложения играет Бот. Поэтому первым этапом в разработке авторизации через Telegram является создание бота.

Для создания и настройки ботов в Telegram есть свой бот – @botfather. В списке команд этого бота необходимо выбрать /newbot, что запустит процесс создания бота. В ответ бот попросит отправить название создаваемого бота, а после, уникальное имя, которое должно заканчиваться на слово Bot. Процесс создания бота для разрабатываемого сервиса представлен на рисунке 3.25.



Рисунок 3.25 – Процесс создания бота

В ответ на введенные данные, бот сообщает что бот успешно создан и присылает токен для доступа к API (рисунок 3.26).

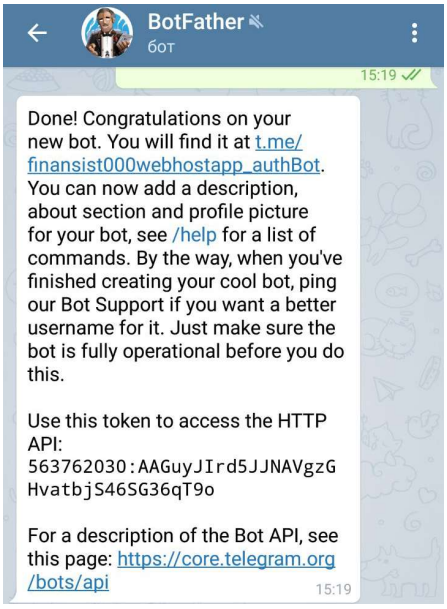


Рисунок 3.26 – Токен для доступа к API.

Данный токен секретная информация и должен быть должным образом защищен от компрометации.

Далее необходимо задать для бота домен. Виджет авторизации будет запущен только на указанном домене. Для задания домена существует команда /setdomain. В ответ бот попросит выбрать бота, для которого необходимо задать домен. После выбора бота, @botfather просит прислать домен разрабатываемого веб-сервиса. Если формат ссылки корректен, то боту присваивается домен (рисунок 3.27).

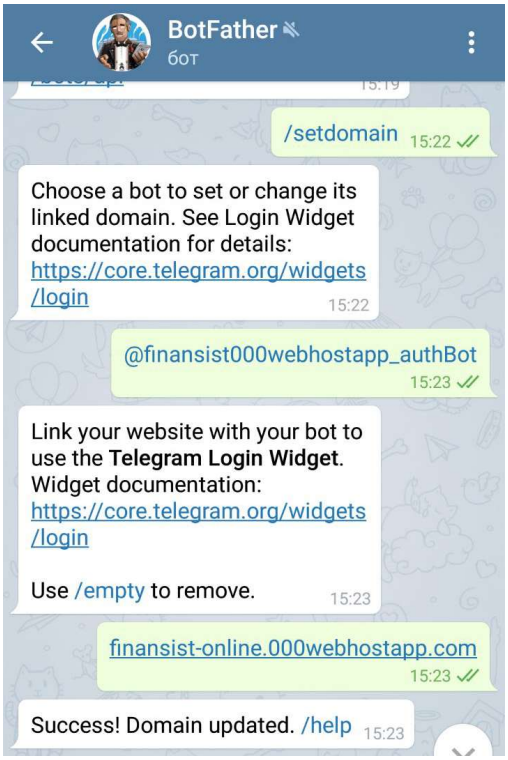


Рисунок 3.27 – Процесс задания домена для бота

На этом процесс создания и настройки бота можно считать завершенным и можно приступить к разработке механизма авторизации.

Настроить виджет можно на веб-сайте Telegram. На специальной странице (рисунок 3.28) указывается имя созданного бота, размеры виджета, внешний вид, задается ссылка для переадресации после авторизации. После указания всех необходимых настроек разработчик получает готовый код, для вставки виджета в заготовленное место на странице.

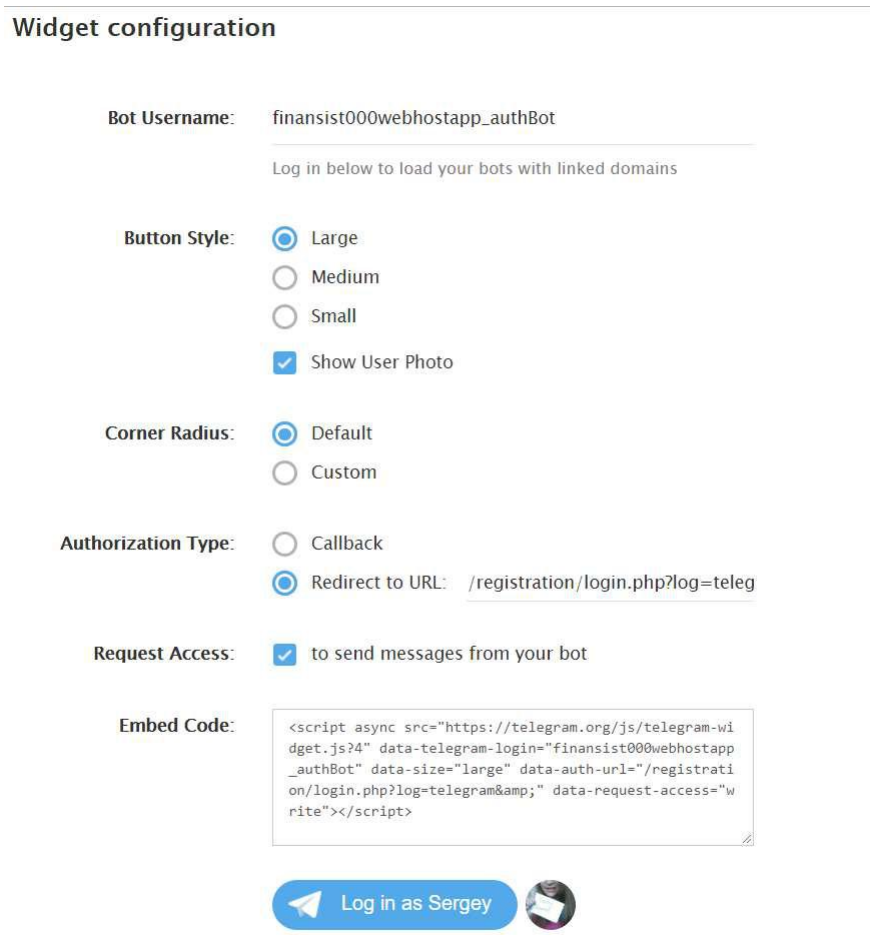


Рисунок 3.28 – Конфигурация виджета авторизации

Виджет авторизации через Telegram на странице веб-сервиса представлен на рисунке 3.29.

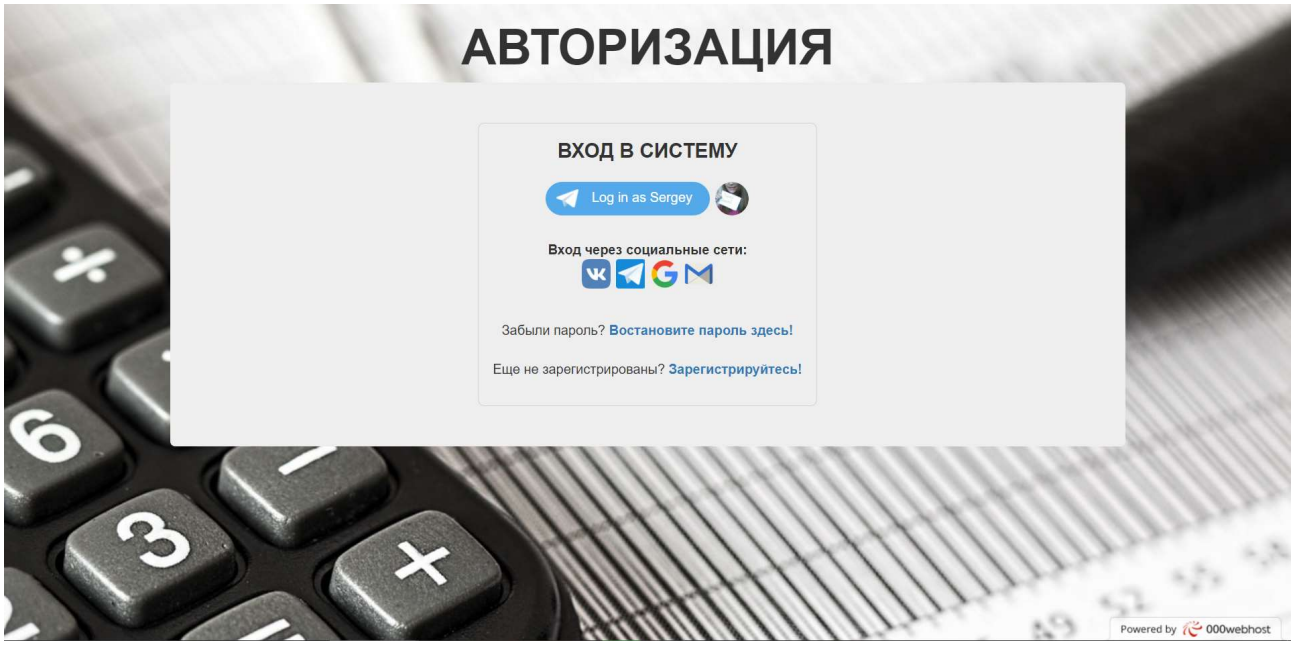


Рисунок 3.29 – Виджет авторизации через Telegram

После авторизации со стороны Telegram пользователь перенаправляется на адрес, указанный в redirect. В GET-массиве содержится вся предоставленная информация о пользователе. Проверить полученные данные на подлинность позволяет функция checkTelegramAuthorization(). Функция принимает аргументом GET-массив и возвращает значение true если данные действительно предоставлены Telegram и никем не модифицированы. Для работы функции необходимо объявить в документе переменную BOT_TOKEN, значением которой является токен, полученный при создании бота.

Если полученные данные корректны, то можно приступить к процессу авторизации на веб-сервисе. Данный процесс полностью аналогичен описанному в конце главы «Авторизация через ВКонтакте».

3.4.3 Авторизация через Google

Из трех разрабатываемых способов авторизации Google имеет самую большую аудиторию пользователей. Для авторизации через Google будет также использован протокол OAuth 2.0.

Прежде всего необходимо создать новый проект в консоли разработчика. Для проекта необходимо указать имя и нажать кнопку «Создать» (рисунок 3.30), в результате чего проект будет создан.

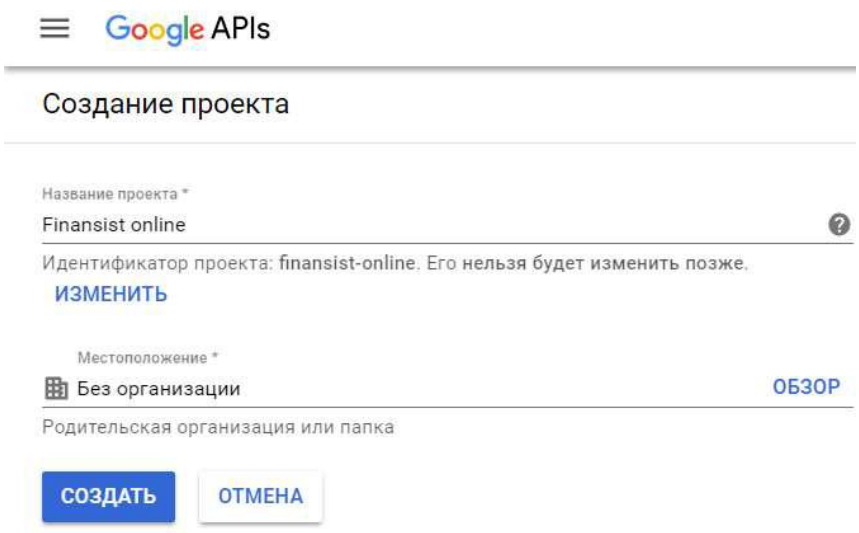


Рисунок 3.30 – Создание проекта в консоли разработчика

Далее необходимо настроить созданный проект. Во вкладке учетные записи нужно указать данные для окна запроса доступа OAuth. Это окно, где пользователь подтверждает свое согласие на предоставление информации веб-сервису. Нужно задать название продукта, указать адрес главной страницы, а также логотип продукта. Настройки для разрабатываемого веб-сервиса представлены на рисунке 3.31.

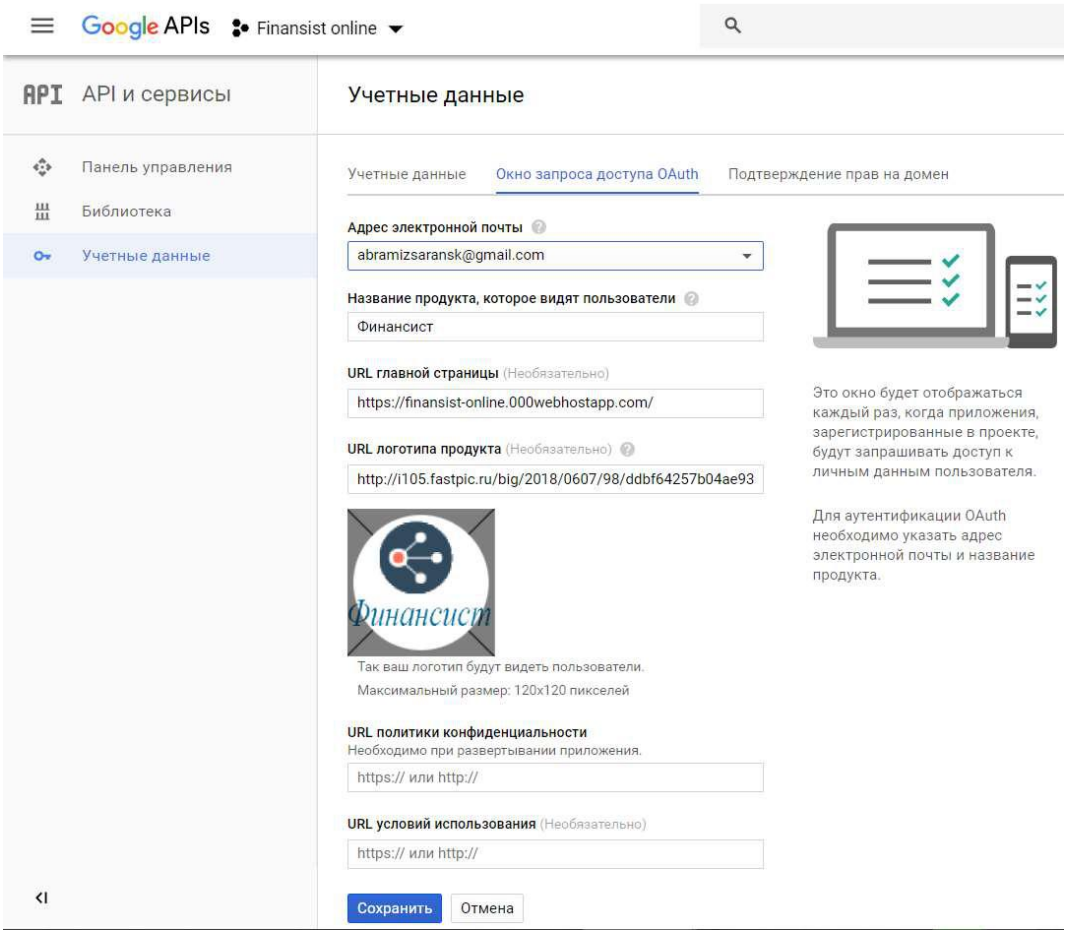


Рисунок 3.31 – Настройки окна запроса доступа OAuth

Далее необходимо создать идентификатор клиента OAuth для созданного проекта. По идентификатору клиента Google будет идентифицировать сервис, запрашивающий доступ к данным. Создать идентификатор можно в вкладке «Учетные записи».

При создании идентификатора нужно выбрать тип приложения. Разрабатываемый веб-сервис имеет тип веб-приложение. Для веб-приложения необходимо задать название, а также указать ограничения.

Ограничения служат для проверки того, что запрос данных осуществляется с разрешенного источника. В разрешенные источники необходимо вписать все адреса, с которых будет осуществляться запрос данных, а в разрешенные перенаправления – адреса на которые разрешено переадресовывать пользователя после авторизации. После заполнения и сохранения всех данных будет создан идентификатор пользователя OAuth. Он состоит из двух частей: идентификатор клиента и секрет клиента. Идентификатор клиента и ограничения для разрабатываемого сервиса представлены на рисунке 3.32.

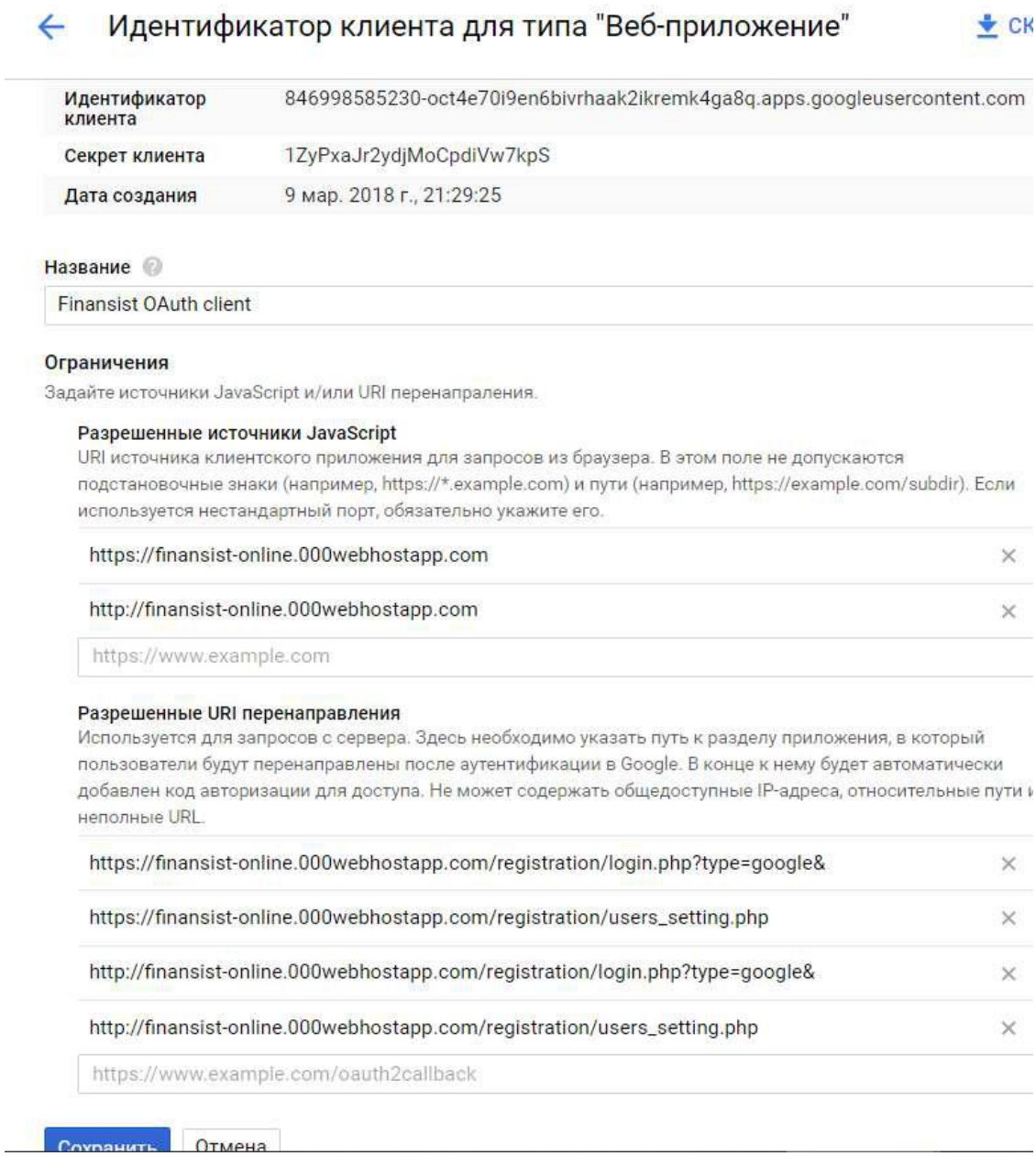


Рисунок 3.32 – Идентификатор клиента для веб-сервиса

На этом все необходимые настройки в консоли разработчика выполнены, можно приступить к реализации механизма авторизации.

Для отображения на странице кнопки авторизации через Google в коде страницы в теге head нужно подключить JavaScript-библиотеку Google Platoform, а также добавить метатег google-signin-client_id. Этот тег хранит идентификатор клиента OAuth.

```
<meta name="google-signin-client_id" content="846998585230-
oct4e70i9en6bivrhaak2ikremk4ga8q.apps.googleusercontent.com">
<script src="https://apis.google.com/js/platform.js" async
defer></script>
```

В месте, где нужно отобразить кнопку, нужно добавить блок div с классом *g-signin2*. В этом блоке задаются все параметры отображаемой кнопки: размер, функция для вызова в случае успешной авторизации, тема для отображения кнопки.

```
 <div class="g-signin2" data-width="250"  data-onsuccess="onSignIn" data-
theme="dark"></div>
```

Вид кнопки на странице авторизации представлен на рисунке 3.33.

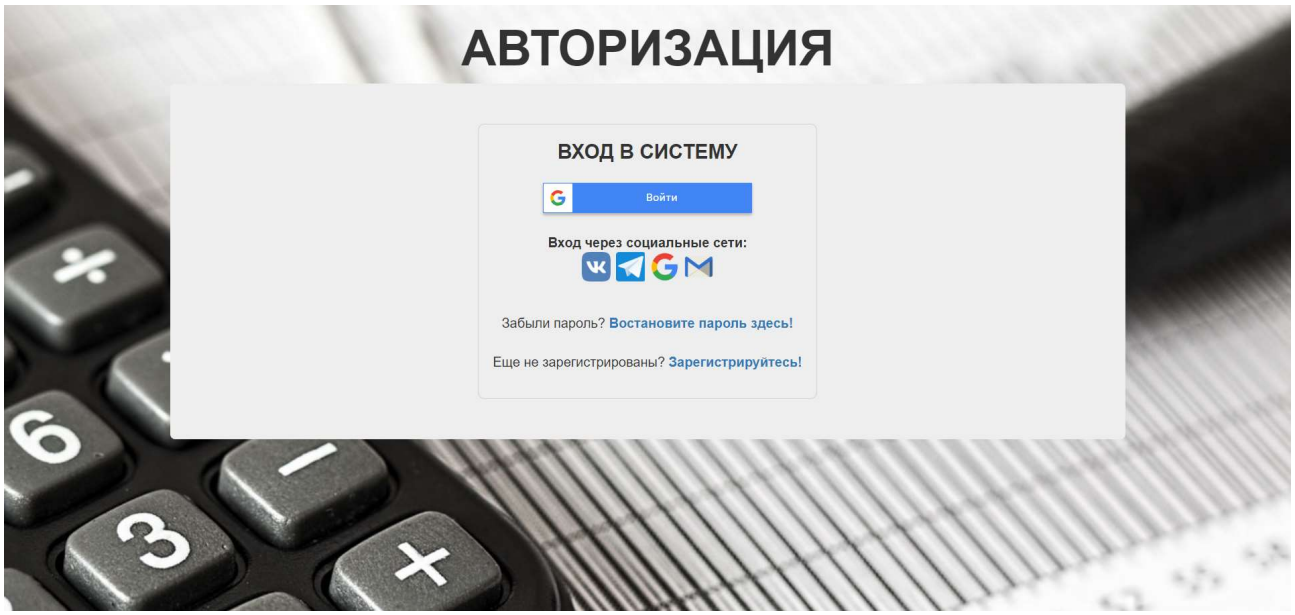


Рисунок 3.33 – Кнопка для авторизации через Google

Если пользователь успешно авторизовался, то вызывается JavaScript-функция onSignIn(). Аргументом функции является полученный от Google объект googleUser. С помощью метода getBasicProfile() создается объект profile,

содержащий в себе всю полученную информацию о пользователе, а с помощью метода getAuthResponse() получается токен, служащий для проверки всех полученных данных на достоверность. Функция onSignIn() создает XML-запрос к скрипту login.php, содержащий в массиве POST токен, полученный от социальной сети [7]. В скрипте login.php находятся несколько функций, осуществляющих процесс авторизации пользователя. Если полученный ответ на запрос «auth_ok», то пользователь считается авторизованным, и функция onSignIn() переадресует пользователя на главную страницу. В случае если ответ «auth_not_found», то пользователь не был найден в базе данных и его необходимо переадресовать на страницу регистрации. В параметрах ссылки для переадресации указываются полученные данные о пользователе, полезные для регистрации. Получение конкретных данных происходит при помощи методов объекта profile:

- getName() – метод для получения имени пользователя;
- getEmail() – получение адреса электронной почты;
- getId() – получение идентификатора пользователя.

Псевдонимом пользователя берется исходя из адреса электронной почты путем обрезки по символ @, но может быть изменен пользователем в процессе регистрации [4].

Полный код функции onSignIn():

```
function onSignIn(googleUser) {
var profile = googleUser.getBasicProfile();
var id_token = googleUser.getAuthResponse().id_token;
      var xhr = new XMLHttpRequest();
      xhr.open('POST', './login.php');
      xhr.setRequestHeader('Content-Type', 'application/x-www-form-
urlencoded');
      xhr.onload = function() {
      if (xhr.responseText=="auth_ok") {
            signOut();
            document.location.replace("./");}
      else
      {
            console.log(xhr.responseText);
            signOut();
document.location.replace("./register.php?reg=google&full_name="+profile.
getName()+"&email="+profile.getEmail()+"&nickname="+profile.getEmail().sp
lit('@',1)+"&id="+profile.getId()+"google");}};
```

```
        xhr.send('idtoken=' + id_token+'&log=google');
}
```

Процесс авторизации пользователя проводится на стороне сервера. Как было описано ранее, функция onSignIn() отправляет XML-запрос к скрипту login.php, содержащий в массиве POST токен, полученный от социальной сети. Для авторизации пользователя необходимо получить идентификатор пользователя социальной сети. Можно было бы также вложить его в запрос, но идентификатор может быть изменен злоумышленниками, поэтому наиболее безопасным способом является его получение из токена.

Для работы с токеном с использованием функций, разработанных компанией Google, необходимо скачать и подключить библиотеку Google API. Все самые свежие версии библиотеки расположены в официальном репозитории google-api-php-client на сайте github.com. Скаченный файл, содержащий библиотеку нужно распаковать в любой каталог на сервер и библиотека готова к подключению. Подключение к библиотеке осуществляется через скрипт autoload.php, расположенный внутри библиотеки:

```
require_once '../libs/google-api-php-client/vendor/autoload.php';
```

Получение идентификатора пользователя из токена осуществляется при помощи функции getUserFromToken(). Сначала проверяется токен на подлинность. Для этого запускается новый Google-клиент. Клиент имеет собственный метод для проверки подлинности токена – verifyIdToken(), принимающий в качестве атрибута токен. Если токен успешно проверен и является подлинным, то вся полученная из токена информация записывается в ассоциативный массив $ticket. Среди этой информации есть искомый идентификатор пользователя, доступ к которому можно получить по ключу sub. Полный код функции getUserFromToken():

```
function getUserFromToken($token) {
      $client = new Google_Client();
      $ticket = $client->verifyIdToken($token);
      if ($ticket) {
            return $ticket['sub']; // user ID
      }
      return false;
}
```

В результате выполнения функции getUserFromToken() полученный идентификатор пользователя сохраняется в переменную $idgoogleuser. Далее с помощью функции login_social_user() происходит поиск пользователя в базе данных и в случает успеха он авторизуется, а скрипт login.php возвращает «auth_ok». Если пользователь не найден, то данные о социальной сети сохраняются в таблицу not_end_registration, а скрипт возвращает сообщение «auth_not_found».

3.5 Реализация расчета кредитного лимита

Задача вычисления кредитного лимита намного сложнее кредитного калькулятора. Для его вычисления необходимо вводить, сохранять и учитывать множество финансовых показателей, такие как: внеоборотные и оборотные активы, капитал и резервы, долгосрочные и краткосрочные обязательства [8].

В рамках данной выпускной работы была разработана структура базы данных компаний, реализован интерфейс для ввода и сохранения этих данных. Сам расчет кредитного лимита на основе сохраненных и вычисляемых параметров - задача для дальнейшей работы.

Для вычисления кредитного лимита пользователь должен создать группу связанных заемщиков, при этом он должен быть авторизован, так как каждая группа, а также компания должны иметь владельца. Создать группу связанных заемщиков можно на странице «Группы связанных заемщиков». Для каждой группы необходимо задать название и описание. Пользователь может создать любое количество групп. Все его группы будут отображены списком (рисунок 3.34.). На этой же странице, нажав на кнопку «Изменить», пользователь может изменить любую свою группу, либо удалить ее, нажав кнопку «Удалить». При удалении группы связанных заемщиков пользователь должен также подтвердить свое желание удалить группу, так как благодаря установленным ограничениям в базе данных при удалении группы связанных заемщиков также удалятся все компании, находящиеся в этой группе, и эта операция необратима.

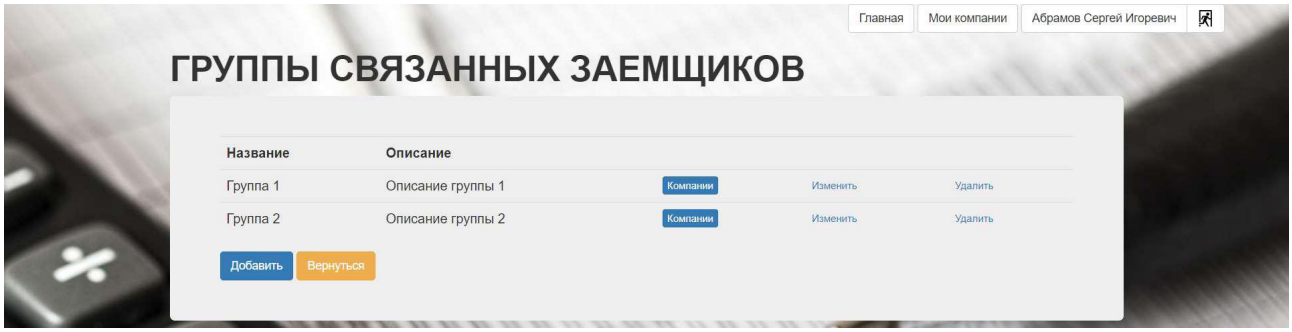


Рисунок 3.34 – Страница «Группы связанных заемщиков»

Нажав на кнопку «Компании», пользователь может просмотреть список компаний для каждой группы. Здесь же можно добавить компанию в группу связанных заемщиков. Для этого пользователю необходимо нажать кнопку «Добавить», заполнить информацию о компании и нажать кнопку «Сохранить». Форма для добавления компании представлена на рисунке 3.35.

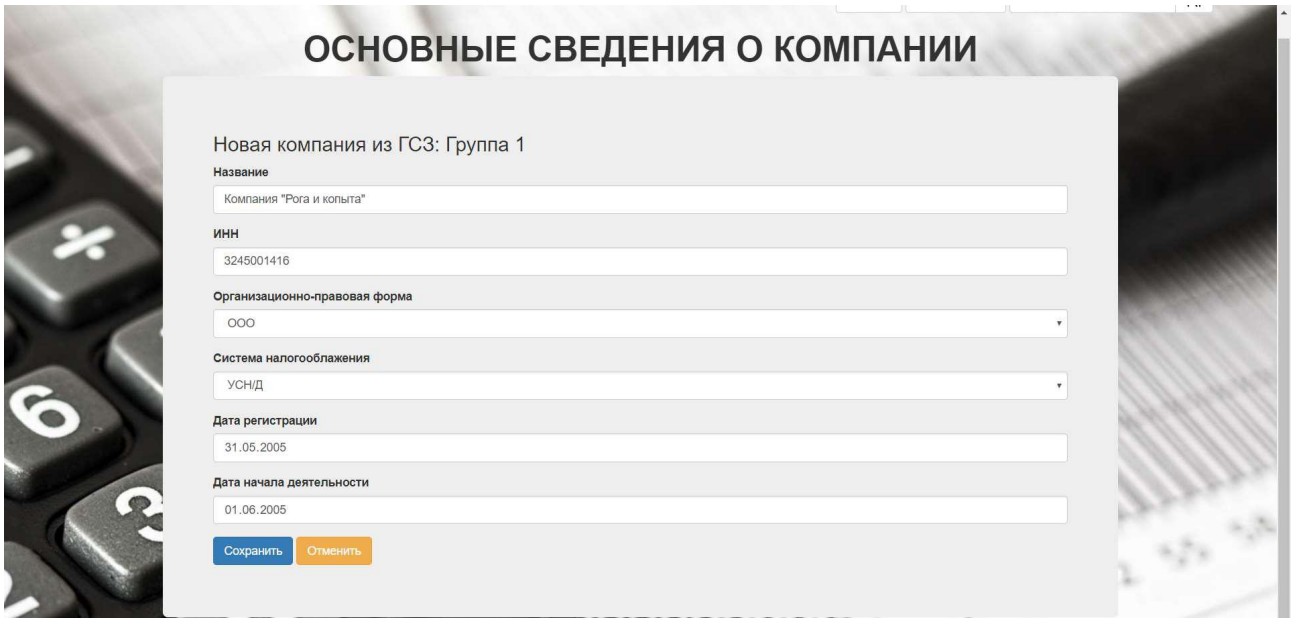


Рисунок 3.35 – Форма для добавления компании в ГСЗ

При нажатии на кнопку «Сохранить» при помощи jQuery отменяется событие submit() и проводится проверка введенных значений. Так, например, дата начала деятельности компании не может быть раньше, чем дата регистрации. А дата регистрации, в свою очередь, не может быть позже текущей даты. Также проверяется длина введенного ИНН при помощи функции is_valid_INN(). Корректная длина ИНН зависит от выбранной организационно-

правовой формы. Для индивидуальных предпринимателей длина ИНН 12, а для остальных 10. При загрузке страницы добавления компании в ГСЗ допустимая длина ИНН, полученная из базы данных, записывается в атрибут *data-INN_Length* для каждого значения выпадающего списка организационно-правой форм. Функция is_valid_INN() сравнивает длину введенного ИНН с значением *data-INN_Length* для выбранной формы. В случае, если даты или ИНН заполнены неверно, пользователю отображаются сообщения с информацией о ошибке (рисунок 3.36).

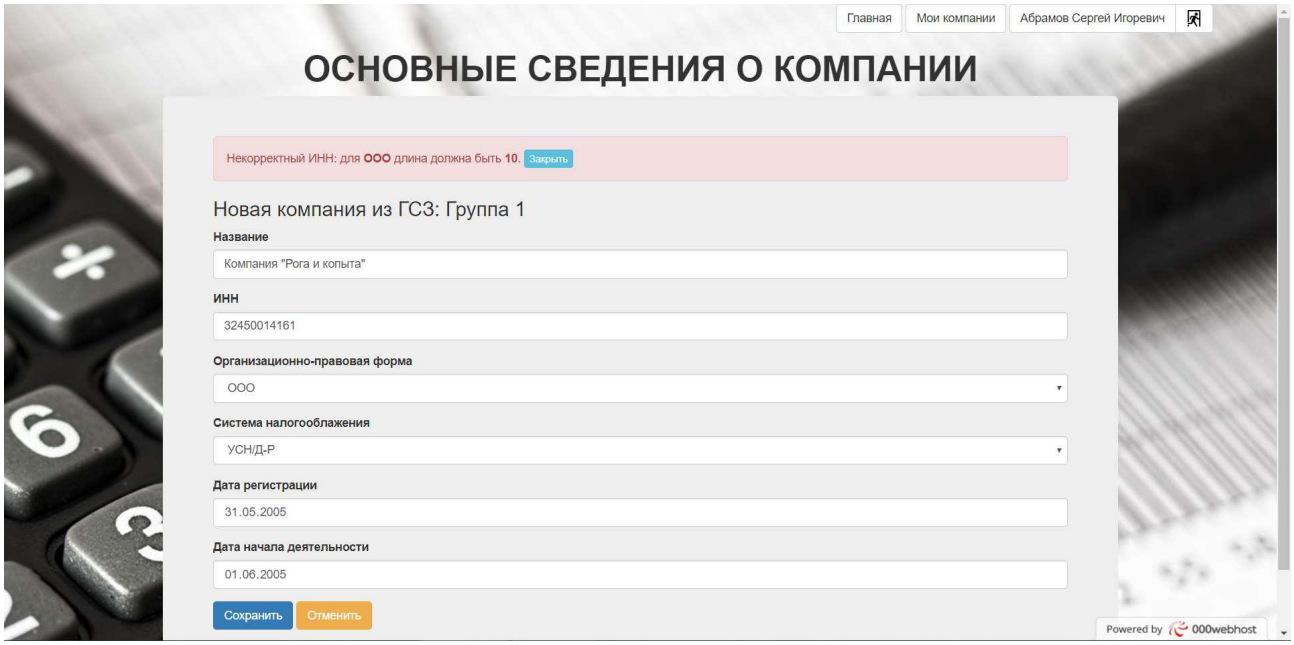


Рисунок 3.36 – Форма для добавления компании с неверным значением ИНН

Также реализованы проверки, не влияющие на отправку формы. Если компания работает менее 6 месяцев или использует систему налогообложения ЕНВД, то пользователь получит сообщение, что компания в расчете кредитного лимита участвовать не будет. После подтверждения пользователем данной информации форма отправляется функцией submit().Для работы с компаниями, а именно их добавления, изменения и удаления реализован скрипт company_save_item.php. При вызове скрипт получает параметр action, в зависимости от которого выполняет необходимое действие. После добавления компании пользователь снова попадает в список компаний, относящихся к

выбранной группе связанных заемщиков (рисунок 3.37). Здесь, также, как и в группах связанных заемщиков, пользователь может удалить или изменить компании.

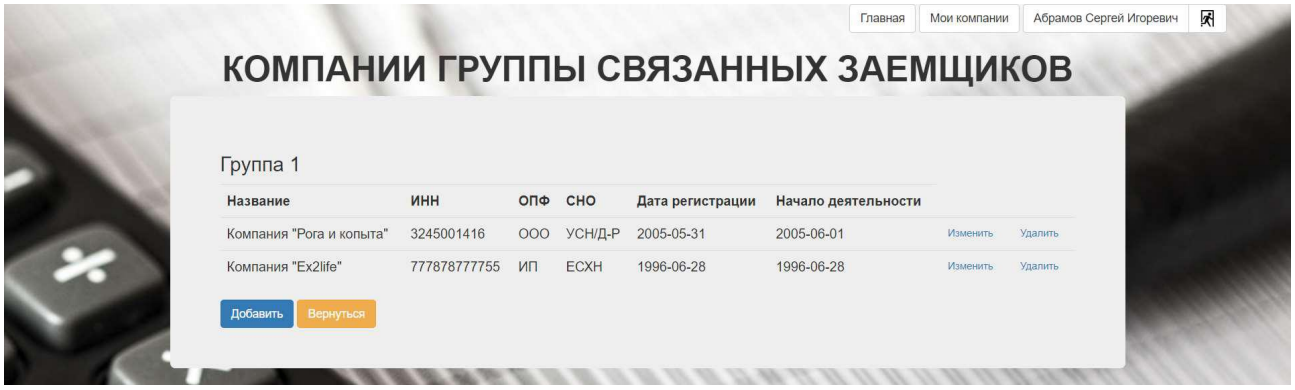


Рисунок 3.37 – Список компаний из «Группа 1»

После добавления группы связанных заемщиков и компаний пользователь может перейти к заполнению финансовых данных. Перейдя по ссылке «Финансовые данные ГСЗ», пользователь попадает на страницу, где представлены все принадлежащие ему группы связанных заемщиков, в которых есть компании. Для каждой группы указано количество копаний в группе, дата расчета лимита, а также начало деятельности группы (самая ранняя дата начала деятельности среди всех компаний, входящих в эту группу). Если дата расчета лимита не задавалась ранее, то для этой группы устанавливается текущая дата. По нажатию на кнопку «Изменить», пользователь может изменить дату расчета лимита. Вид страницы «Финансовые данные групп связанных заемщиков представлен на рисунке 3.38.

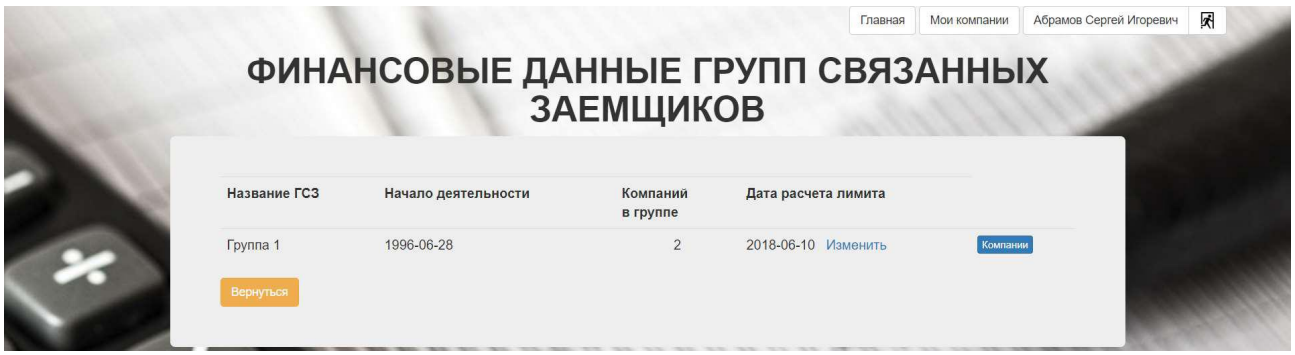


Рисунок 3.38 – Финансовые данные групп связанных заемщиков

Нажав, на кнопку «Компании» пользователь веб-сервиса перейдет в список компаний для выбранной группы (рисунок 3.39).

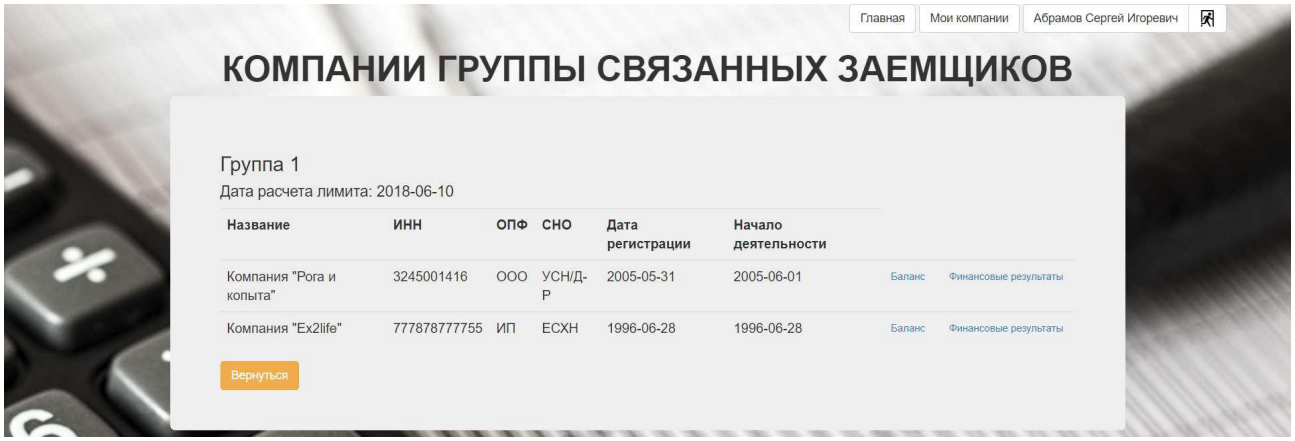


Рисунок 3.39 – Список компаний для выбранной группы

Кнопка «Баланс» открывает форму, содержащую основную информацию о компании, а также кнопки для заполнения или просмотра баланса. Для организаций даты определения баланса отсчитываются от начала текущего квартала, а именно:

1) За 6 месяцев до начала текущего квартала.

2) За 3 месяца до начала текущего квартала.

3) Дата начала текущего квартала.

А для индивидуальных предпринимателей даты определения баланса отсчитываются от даты расчета лимита:

1) За 6 месяцев до даты расчета лимита.

2) За 3 месяцев до даты расчета лимита.

3) Дата расчета лимита.

Код функции, осуществляющей расчет дат определения баланса:

```
function get_Balance_Dates($Date_calc_limit, $is_Corporation = 0){
      $Balance_Dates = [];
      $Date1 = new DateTime($Date_calc_limit);
      if ($is_Corporation)
      {//Для организации
            $month = (int)($Date1->format("n"));
            if ($month <= 3) $Date3 = $Date1->format("Y-01-01");
            if (($month >= 4) and ($month <= 6)) $Date3 = $Date1-
>format("Y-04-01");
            if (($month >= 7) and ($month <= 9)) $Date3 = $Date1-
>format("Y-07-01");
```

```
            if ($month >= 10) $Date3 = $Date1->format("Y-10-01");
      }
      else {//Для ИП
$Date3 = $Date_calc_limit;
      };
      $Date = new DateTime($Date3);
      $Date->modify("-6 month");
      $Balance_Dates[] = $Date->format("Y-m-d");
      $Date = new DateTime($Date3);
      $Date->modify("-3 month");
      $Balance_Dates[] = $Date->format("Y-m-d");
      $Balance_Dates[] = $Date3;
      return $Balance_Dates;
}
```

Страница «Баланс организации» представлена на рисунке 3.40.

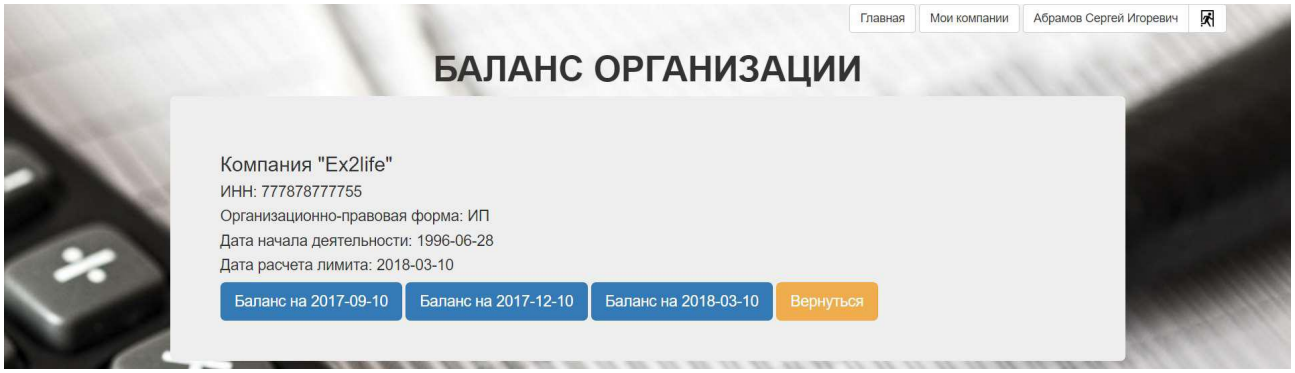


Рисунок 3.40 – Страница «Баланс организации»

Для заполнения или просмотра баланса организации на определенную дату пользователю необходимо нажать одну из кнопок. В результате будет открыта страница balance_form.php, содержащая все необходимые поля для заполнения баланса. Основная информация о каждой статье баланса, это ее код, описание и сумма баланса для статьи. Все статьи баланса хранятся в базе данных и загружаются в момент запроса страницы при помощи функции get_Corporation_Balance_Part(). Эта функция осуществляет формирование массива, состоящего из статей баланса, для отображения пользователю. В зависимости от значения параметра Balance_Part функция возвращает либо статьи баланса «Актив», либо «Пассив».

Сначала в базе данных осуществляется поиск данных баланса компании за выбранную дату. Если информация найдена, то заголовки статей баланса и сохраненные суммы баланса загружаются из таблицы Corp_Balance_Results, а

если не найдена, то из таблицы Corp_Balance_Articles загружаются заголовки статей для баланса с нулевыми значениями сумм баланса.

Далее каждый раздел обрабатывает отдельно с помощью цикла. Раздел состоит из родительских и дочерних статей. С помощью вложенного цикла проверяется каждая статья, и если статья является дочерней статьей, то для удобного отображения пользователю ей устанавливается отступ вправо на два пробела и ей задается загруженное значение, которое пользователь сможет изменять. А если статья является родительской, т.е. состоит из нескольких дочерних статей, то для такой статьи вычисляется сумма значений всех статей, входящих в нее. Эта сумма будет отображена пользователю, и он не сможет ее изменять.

В конце каждого цикла, обрабатывающего раздел, вычисляется сумма всех статей, которые входят в этот раздел.

И окончательным этапом работы функции является вычисление суммы всех статей, входящих либо в «Актив» баланса, либо в «Пассив».

Вся загруженная с помощью этой функции информация отображается пользователю в виде таблицы (рисунок 3.41).

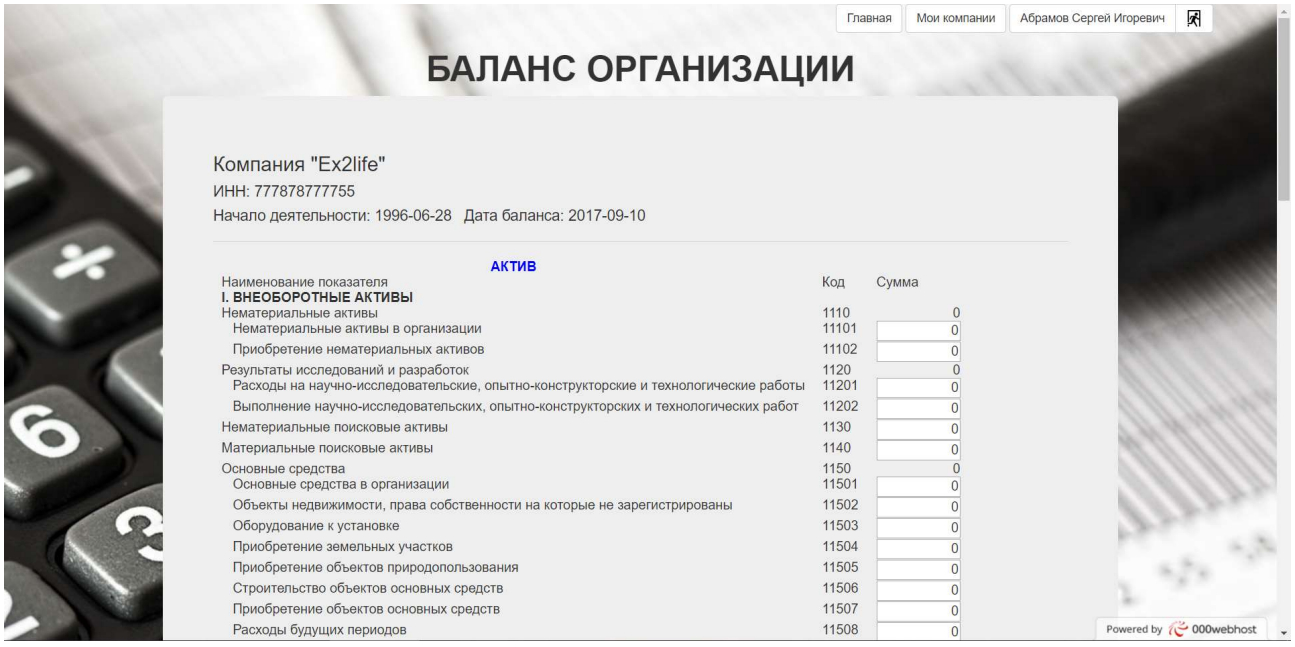


Рисунок 3.41 – Форма для заполнения баланса организации на определенную дату

ЗАКЛЮЧЕНИЕ

В настоящее время широко используются несколько различных операционных систем, что создает трудности при разработке кроссплатформенных приложений. Для того, чтобы приложение было доступно максимальному количеству пользователей, разработчик должен выпусить версии по меньшей мере для Windows, Linux, MacOS, Android, IOS и Windows Phone. В разработке для каждой платформы есть множество своих нюансов, изучение которых повлечет за собой существенное увеличение времени разработки. Именно поэтому сейчас клиент-серверные веб-приложения стремительным темпом вытесняют локальные приложения.

С каждым годом растет скорость доступа к сети интернет, происходит развитие серверных и клиентских технологий. Использование клиент-серверных приложений позволяет сделать интерфейс веб-сервиса, не уступающий локальным приложениям по удобству использования и возможностям. Благодаря низким требованиям к ресурсам клиентского устройства, а также отсутствию зависимости от операционной системы такие веб-сервисы доступны максимальному количеству пользователей с различных десктопных и мобильных устройств.

Целью написания данной бакалаврской работы являлась разработка веб-сервиса для расчета финансовых показателей, применяемых банками при оценке сумм кредитов, выдаваемых организациям и предпринимателям. Самостоятельное использование этого сервиса организацией значительно облегчает расчет графика платежей за кредиты, а также расчет кредитного лимита для групп связанных заемщиков.

В ходе разработки была изучена предметная область, проанализированы требования к сервису. Была спроектирована и разработана база данных, а также функции для работы с ней. Для разрабатываемого веб-сервиса была реализована авторизация и регистрация пользователей (в том числе с помощью популярных социальных сетей), интерфейс и функции клиентской части. Было

написано множество функций и скриптов серверной части для осуществления основных возможностей веб-сервиса.

Реализованный проект будет основным звеном в последующем расширении возможностей сервиса. Так как разработанная система является модульной, она может быть без проблем расширена новыми функциям, а отдельные части проекта, например, система авторизации и регистрации пользователей, могут быть использованы для разработки других веб-сервисов.

В ходе разработки мною был изучен большой объем документации как клиентских, так и серверных технологий. Получены навыки в разработке клиент-серверных приложений и верстке веб-страниц с использованием современных технологий.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Дейт К. Введение в системы баз данных / К. Дейт. – 8-е изд. – М. : Вильямс, 2006. – 1328 с.

2 Димандт Д. jQuery – Javascript нового поколения / Д. Димандт // RSDN Magazine. – 2007. – №1 – С. 56-54

3 Евдокимов А. П. Создание сайтов своими руками на Bootstrap / А. П. Евдокимов, М. В. Финков. – СПб. : Наука и Техника, 2017. – 240 с.

4 Колисниченко Д. Н. PHP и MySQL. Разработка веб-приложений / Д. Н. Колисниченко. – 6-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2017. – 640 с.

5 Котеров Д. В. PHP 5 / Д. В. Котеров, А. Ф. Костарев. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2008. – 1104 с.

6 Кузнецов М. Самоучитель PHP 5/6 / М. Кузнецов, И. Симдянов – 3-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2009. – 672 с.

7 Прохоренок Н. А. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н. А. Прохоренок, В. А. Дронов. – 4-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2015. – 768 с.

8 Степанов В. Расчет максимальной суммы кредита для бизнеса [Электронный ресурс] / В. Степанов // Бизнес-портал «Фабрика манимейкеров». – [Б. м. : б. и.], 2012. – Режим доступа: http://moneymakerfactory.ru/articles/Kak-banki-rasschityvajut-maksimalnuju-summu-kredita/. – Загл. с экрана.

9 Шевчук А. jQuery для начинающих / Хабр [Электронный ресурс] / А. Шевчук // Habrahabr: Habrahabr.ru. – [Б. м. : б. и.], 2008. – Режим доступа: https://habr.com/post/38208/. – Загл. с экрана.

10 Шелдон Р. MySQL 5: базовый курс / Р. Шелдон, Д. Мойе. – М. : Диалектика, 2007. – 880 с.

11 Аннуитетные платежи – расчёт, формула [Электронный ресурс] : ресурс о финансах. – [Б. м. : б. и.], [2017]. – Режим доступа: http://www.platesh.ru/annuitetnie-plateshi/. – Загл. с экрана.

12 Кредит: виды, формы, типы, сущность, функции [Электронный ресурс] : ресурс о кредитах. – [Б. м. : б. и.], [2015]. – Режим доступа: https://v-kredit.net/. – Загл. с экрана.

13 Дифференцированный платеж – расчёт, формула [Электронный ресурс] : ресурс о финансах. – [Б. м. : б. и.], [2017]. – Режим доступа: http://www.platesh.ru/differencirovannie-plateshi/. – Загл. с экрана.

14 Bootstrap 3. Документация на русском языке [Электронный ресурс] : документация для Bootstrap 3. – [Б. м. : б. и.], [2018]. – Режим доступа: http://bootstrap-3.ru/index.php. – Загл. с экрана.

15 PHP: Руководство по PHP - Manual [Электронный ресурс] : руководство по PHP. – [Б. м. : б. и.], [2018]. – Режим доступа: http://www.php.net/manual/ru/. – Загл. с экрана.

ПРИЛОЖЕНИЕ А

(обязательное)

Функции для авторизации и регистрации пользователей

```
<?php
require($_SERVER['DOCUMENT_ROOT'].'/script/connection_params.php');
require('shablon_email.php');
 /*
 * Отправка письма для подтверждения почты
 */
function send_confirm_message($user)
{


$subject="Подтверждение регистрации на сайте Финансист";
$headers = "To: name <".$to.">\r\n";
$headers = "From: Финансист\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: text/html; charset=utf-8";
$user['link']=$_SERVER['HTTP_HOST']."/registration/verification.php?mail=
".$user['email']."&hash=".md5(SALT.$user['id']);
$message = gener_email_html($user);


// Отправляем
mail($user['email'], $subject, $message,$headers);
redirect('thankyou_regist.php');
}

/*
 * Отправка письма для изменения забытого пароля
 */
function send_password_message($user)
{


$subject="Восстановление забытого пароля на сайте Финансист";
$headers = "To: name <".$to.">\r\n";
$headers = "From: Финансист\r\n";
$headers .= "MIME-Version: 1.0\r\n";
$headers .= "Content-Type: text/html; charset=utf-8";
$microtime=microtime();
$dbh = db_connect();
forget_password_add($dbh, $user['email'],$microtime);
db_close($dbh);
$user['link']=$_SERVER['HTTP_HOST']."/registration/verification_pass.php?
mail=".$user['email']."&hash=".md5(SALT.$microtime.SALT.$user['id']);
$message = gener_email_html_pass($user);

// Отправляем
mail($user['email'], $subject, $message,$headers);
redirect('pass_mail_send.php');
```

```
}


/*
 * Процесс подтверждения почты
 */
function good_email($dbh, $array_get)
{
      $id_email=db_user_id_find_by_email($dbh, $array_get['mail']);
      if ($array_get['hash']==md5(SALT.$id_email)) {
            $status=db_user_status_update($dbh, $id_email);
            return true;
      } else {
            return false;
      }
}

/*
 * Процесс обновления пароля
 */
function update_forget_pass($dbh, $pass2, $errors)
{
      $errors = empty_errors();
      // считываем строки из запроса
      read_string($_POST, 'password', $pass2, $errors, 6, 24, true);
      read_string($_POST, 'password_confirmation', $pass2, $errors, 6,
24, true);

      if (has_errors($errors))
            return false;

      // форма передана правильно, проверяем hash

      $microtime=forget_password_find_time($dbh,$pass2['mail']);
      $id_mail=db_user_id_find_by_email($dbh, $pass2['mail']);
      if (md5(SALT.$microtime.SALT.$id_mail)!=$pass2['hash'])
      {
            forget_password_del($dbh,$pass2['mail']);
            return false;
      }

      // пароль и подтверждение пароля должны совпадать
      if (!is_error($errors, 'password') &&
                  !is_error($errors, 'password_confirmation') &&
                  $pass2['password'] != $pass2['password_confirmation']) {
            $errors['fields'][] = 'password';
            add_error($errors, 'password_confirmation', 'dont-match');
      }

      if (has_errors($errors))
            return false;
      $db_user = db_user_find_by_login($dbh, $pass2['mail']);
      // защищаем пароль пользователя
      $db_user['password'] = password_hash($pass2['password'],
PASSWORD_DEFAULT);
```

```
      unset($pass2['password']);
      unset($pass2['password_confirmation']);

      // форма передана правильно, обновляем пароль пользователя в базе
данных
      $db_user = db_user_update($dbh, $db_user);
      forget_password_del($dbh,$pass2['mail']);
      return true;
}

 /*
 * Выполняет вывод указанного шаблона $template с данными
 */
function render($template, $data=array())
{
      extract($data);
      require('templates/'.$template.'.php');
}

/*
 * Выполняет переадресацию на указанный адрес $url
 */
function redirect($url)
{
      session_write_close();
      header('Location: '.$url);
      exit;
}

/* ----------------------------------------------------------------------
------
 * Функции работы с массивом ошибок
 */

/*
 * Инициализирует структуру массива для хранения информации об ошибках
 */
function empty_errors()
{
      return array(
            'fields'   => array(),
            'messages' => array(),
      );
}

/*
 * Проверяет, что есть ошибочные поля в описании ошибок
 */
function has_errors($errors)
{
      return isset($errors['fields']) && count($errors['fields']) > 0;
}

/*
 * Проверяет, что указанное поле есть в списке ошибочных полей
```

```
 */
function is_error($errors, $field)
{
     return isset($errors['fields']) && in_array($field,
$errors['fields']);
}

/*
 * Добавляет описание ошибки в массив ошибок
 */
function add_error(&$errors, $field, $description)
{
     $errors['fields'][] = $field;
     $errors['messages'][$field] = "@$field-$description";
     return false;
}


/* ------------------------------------------------------------------------
------
 * Валидация данных
 */

/*
 * Проверяет корректность строки в форме, если строка корректна, копирует
ее в $obj
 * и возвращает true или false и заполненный массив ошибок, если нет
 */
function read_string($form, $field, &$obj, &$errors, $min, $max,
$is_required, $default=null, $trim=true)
{
     $obj[$field] = $default;
     if (!isset($form[$field])) {
          return $is_required ? add_error($errors, $field, 'required') :
true;
     }

     $value = $trim ? trim($form[$field]) : $form[$field];
     if ($value == '' && $is_required)
          return add_error($errors, $field, 'required');

     if (strlen($value) < $min)
          return add_error($errors, $field, 'too-short');

     if (strlen($value) > $max)
          return add_error($errors, $field, 'too-long');

     $obj[$field] = $value;
     return true;
}

/*
 * Проверяет корректность адреса электронной почты в форме, если адрес
корректен,
```

```
 * копирует его в $obj и возвращает true или false и заполненный массив
ошибок, если нет
 */
function read_email($form, $field, &$obj, &$errors, $min, $max,
$is_required, $default=null)
{
      $obj[$field] = $default;
      if (!isset($form[$field])) {
            return $is_required ? add_error($errors, $field, 'required') :
true;
      }

      $value = trim($form[$field]);
      if (strlen($value) < $min)
            return add_error($errors, $field, 'too-short');

      if (strlen($value) > $max)
            return add_error($errors, $field, 'too-long');

      // проверяем, что в строке задан адрес электронной почты
      if (!filter_var($value, FILTER_VALIDATE_EMAIL))
            return add_error($errors, $field, 'invalid');

      $obj[$field] = $value;
      return true;
}

/*
 * Проверяет корректность выбора одного из значений в форме, если выбрано
значение
 * из указанного списка, копирует его в $obj и возвращает true или false
и заполненный
 * массив ошибок, если нет
 */
function read_list($form, $field, &$obj, &$errors, $list, $is_required,
$default=null)
{
      $obj[$field] = $default;
      if (!isset($form[$field])) {
            return $is_required ? add_error($errors, $field, 'required') :
true;
      }

      $value = trim($form[$field]);
      if (!in_array($value, $list))
            return add_error($errors, $field, 'invalid');

      $obj[$field] = $value;
      return true;
}

/*
 * Проверяет корректность логического значения, если корректно, копирует
его
```

```
 * в $obj и возвращает true или false и заполненный массив ошибок, если
нет
 */
function read_bool($form, $field, &$obj, &$errors, $true, $is_required,
$default=null)
{
      $obj[$field] = $default;
      if (!isset($form[$field])) {
            return $is_required ? add_error($errors, $field, 'required') :
true;
      }

      $value = trim($form[$field]);
      $obj[$field] = $value === $true;
      return true;
}


/* ----------------------------------------------------------------------
------
 * Текущий пользователь, для вошедшего в систему пользователя мы храним
 * в сессии его идентификатор в базе данных
 */

/*
 * Возвращает идентификатор пользователя, выполнившего вход в систему
 */
function get_current_user_id()
{
      return $_SESSION['user_id'];
}

/*
 * Сохраняет идентификатор пользователя, выполнившего вход
 */
function store_current_user_id($id)
{
      $_SESSION['user_id'] = $id;
}

/*
 * Проверяет, что у нас имеется вошедший в систему пользователь
 */
function is_current_user()
{
      return isset($_SESSION['user_id']);
}

/*
 * Сбрасывает идентификатор пользователя, выполнившего вход
 */
function reset_current_user_id()
{
      unset($_SESSION['user_id']);
}
```

```
/*
 * Выполняет вход пользователя в систему, возвращает true, если вход
 * выполнен успешно, и false и заполненный массив ошибок в противном
 * случае
 */
function login_user($dbh, &$user, &$errors)
{
     $user = array();
     $errors = empty_errors();

     // считываем строки из запроса
     read_string($_POST, 'username', $user, $errors, 2, 64, true);
     read_string($_POST, 'password', $user, $errors, 6, 20, true);

     if (has_errors($errors))
          return false;

     // форма передана правильно, ищем пользователя и проверяем пароль
     $db_user = db_user_find_by_login($dbh, $user['username']);
     // смотрим, есть ли такой пользователь и правильно ли передан
пароль
     if ($db_user == null || !password_verify($user['password'],
$db_user['password']))
          return add_error($errors, 'password', 'invalid');
     if ($db_user['status_active']=='0')
     {
          return add_error($errors, 'status_active', 'no_active');
     }
     // пользователь ввел правильные имя и пароль, запоминаем его в
сессии
     store_current_user_id($db_user['id']);
     // проверяем, нужно ли обновить кеш пароля
     if (password_needs_rehash($db_user['password'], PASSWORD_DEFAULT))
     {
    // Кеш пароля нужно обновить, в связи с изменением алгоритма
    $db_user['password'] = password_hash($user['password'],
PASSWORD_DEFAULT);
     // обновим кеш в базе данных
    $db_user = db_user_update($dbh, $db_user);
    // don't forget to store the new hash!
    }
     return true;
}


/*
 * Выполняет вход пользователя в систему через социальные сети,
возвращает true, если вход
 * выполнен успешно, и false и заполненный массив ошибок в противном
 * случае
 */
function login_social_user($dbh, $soc, $socid, $errors)
{
     $errors = empty_errors();
```

```
      // форма передана правильно, ищем пользователя
      $id_otv = db_id_find_by_socid($dbh, $soc, $socid);
      $db_user=db_user_find_by_id($dbh, $id_otv);
      if ($db_user['status_active']=='0')
      {
            return add_error($errors, 'status_active', 'no_active');
      }
      store_current_user_id($id_otv);
      return $id_otv;
}

/*
 * Выполняет выход пользователя из системы
 */
function logout_user()
{
      reset_current_user_id();
}

/*
 * Выполняет регистрацию пользователя, возвращает true, если регистраиция
 * завершилась успешно, и false и заполненный массив ошибок в противном
 * случае
 */
function register_user($dbh, &$user, &$errors)
{
      $user = array();
      $errors = empty_errors();

      // считываем строки из запроса
      read_string($_POST, 'nickname', $user, $errors, 2, 64, true);
      read_email($_POST, 'email', $user, $errors, 2, 64, true);
      read_string($_POST, 'password', $user, $errors, 6, 24, true);
      read_string($_POST, 'password_confirmation', $user, $errors, 6, 24,
true);
      read_string($_POST, 'fullname', $user, $errors, 1, 80, true);
      read_string($_POST, 'socid_soc', $user, $errors, 3, 225, false);
      read_bool($_POST, 'newsletter', $user, $errors, '1', false, false);

      // пароль и подтверждение пароля должны совпадать
      if (!is_error($errors, 'password') &&
                  !is_error($errors, 'password_confirmation') &&
                  $user['password'] != $user['password_confirmation']) {
            $errors['fields'][] = 'password';
            add_error($errors, 'password_confirmation', 'dont-match');
      }

      //проверка уникальности никнейма
      if (!db_freedom_nick_find($dbh, $user['nickname']))
      {
            add_error($errors, 'nickname', 'nick-not-freedom');
      }
      //проверка уникальности адреса электронной почты
      if (!db_freedom_email_find($dbh, $user['email']))
```

```
	{
		add_error($errors, 'email', 'email-not-freedom');
	}

	if (has_errors($errors))
		return false;

	// защищаем пароль пользователя
	$user['password'] = password_hash($user['password'],
PASSWORD_DEFAULT);
	unset($user['password_confirmation']);

	// форма передана правильно, сохраняем пользователя в базу данных
	$db_user = db_user_insert($dbh, $user);
	if (isset($user['socid_soc']))//если пользователь пытался
изначально зайти через социальную сеть
	{
		db_user_socid_insert($dbh, db_user_social_reg_ok_insert($dbh,
$user['socid_soc']), $db_user['id']);

	}
	else
	{
		db_user_socid_insert_null($dbh, $db_user['id']);
	}
	send_confirm_message($db_user);
	return true;
}


/*
 * Выполняет подготовку данных для отправки письма пользователю
 * при востановлении пароля возвращает true, если регистраиция
 * завершилась успешно, и false и заполненный массив ошибок в противном
 * случае
 */
function send_password_mail($dbh, &$user, &$errors)
{
	$user = array();
	$errors = empty_errors();

	// считываем строки из запроса
	read_email($_POST, 'email', $user, $errors, 2, 64, true);

	if (has_errors($errors))
		return false;

	// форма передана правильно, ищем пользователя
	$db_user = db_user_find_by_login($dbh, $user['email']);

	if (!db_freedom_email_find($dbh, $user['email']))
	{
		send_password_message($db_user);
	}
	// форма передана правильно, сохраняем пользователя в базу данных
```

```
      // автоматически логиним пользователя после регистрации, запоминая
его в сессии
      store_current_user_id($db_user['id']);
      return true;
}


/*
 * Выполняет чтение и проверку введенных 3 паролей
 */
function update_pass($dbh, &$pass3, &$errors)
{
      $pass3 = array();
      $errors = empty_errors();

      // считываем строки из запроса
      read_string($_POST, 'old_password', $pass3, $errors, 6, 24, true);
      read_string($_POST, 'password', $pass3, $errors, 6, 24, true);
      read_string($_POST, 'password_confirmation', $pass3, $errors, 6,
24, true);


      // форма передана правильно, ищем пользователя и проверяем пароль
      $db_user = db_user_find_by_id($dbh, get_current_user_id());
      // смотрим, есть ли такой пользователь и правильно ли передан
пароль
      if ($db_user == null || !password_verify($pass3['old_password'],
$db_user['password']))
            return add_error($errors, 'old_password', 'invalid');
      // пользователь ввел правильные имя и пароль
      unset($pass3['old_password']);
      // пароль и подтверждение пароля должны совпадать
      if (!is_error($errors, 'password') &&
                  !is_error($errors, 'password_confirmation') &&
                  $pass3['password'] != $pass3['password_confirmation']) {
            $errors['fields'][] = 'password';
            add_error($errors, 'password_confirmation', 'dont-match');
      }

      if (has_errors($errors))
            return false;

      // защищаем пароль пользователя
      $db_user['password'] = password_hash($pass3['password'],
PASSWORD_DEFAULT);
      unset($pass3['password']);
      unset($pass3['password_confirmation']);

      // форма передана правильно, обновляем пароль пользователя в базе
данных
      $db_user = db_user_update($dbh, $db_user);
      // автоматически выходим из профиля
      logout_user();
```

```
      return true;
}

/*
 * Выполняет проверку и обновление новых данных пользователя
 */
function update_infoprofile($dbh, &$new_info, &$errors)
{
      $new_info = array();
      $errors = empty_errors();

      // считываем строки из запроса
      read_string($_POST, 'nickname', $new_info, $errors, 2, 64, true);
      //read_email($_POST, 'email', $new_info, $errors, 2, 64, true);
      read_string($_POST, 'fullname', $new_info, $errors, 1, 80, true);
      read_bool($_POST, 'newsletter', $new_info, $errors, '1', false,
false);

      // форма передана правильно, ищем пользователя
      $db_user = db_user_find_by_id($dbh, get_current_user_id());
      if (($db_user['nickname']!=$new_info['nickname']) and
!db_freedom_nick_find($dbh, $new_info['nickname']))
      {
            add_error($errors, 'nickname', 'nick-not-freedom');
      }
      if (has_errors($errors))
            return false;
      $db_user['nickname']=$new_info['nickname'];
      //$db_user['email']=$new_info['email'];
      $db_user['fullname']=$new_info['fullname'];
      $db_user['newsletter']=$new_info['newsletter'];

      unset($new_info);

      // форма передана правильно, обновляем пароль пользователя в базе
данных
      $db_user = db_user_update($dbh, $db_user);
      return true;
}


/* ------------------------------------------------------------------------
------
 * Список пользователей в базе данных
 */

/*
 * Выполняет подключение к базе данных
 */
function db_connect()
{
      $dbh = mysqli_connect(MYSQL_SERVER, MYSQL_USER, MYSQL_PASSWORD,
MYSQL_DB);

      // проверка соединения
```

```
      if (mysqli_connect_errno())
            db_handle_error($dbh);

      mysqli_set_charset($dbh, "utf8");

      return $dbh;
}

/*
 * Закрывает подключение к базе данных
 */
function db_close($dbh)
{
      mysqli_close($dbh);
}

/*
 * Обработка ошибок подключения к базе данных
 */
function db_handle_error($dbh)
{
      $code = '@unknown-error';
      $message = '';
      if (mysqli_connect_error()) {
            $code = '@connect-error';
            $message = mysqli_connect_error();
      }

      if (mysqli_error($dbh)) {
            $code = '@query-error';
            $message =mysqli_error($dbh);
      }

      render('error', array(
            'code' => $code, 'message' => $message,
      ));
      exit;
}

/*
 * Извлекает из базы компании текущего пользователя
 */
function db_company_find_all_for_current_user($dbh, $id)
{
      $query = 'SELECT company.Id, company.Date_Registr,
company.Date_Begin_Work, company.Name, company.INN, OPF.Brief_Name as
opf_brief_name, OPF.Full_Name as opf_full_name, SNO.Brief_Name as
sno_brief_name, SNO.Full_Name as sno_full_name FROM company, OPF, SNO
WHERE company.SNO_Id=SNO.Id and company.OPF_Id=OPF.Id and user_id=?';
     // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);

      mysqli_stmt_bind_param($stmt, 's', $id);
```

```
	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
	if ($qr === false)
		db_handle_error($dbh);
	// последовательно извлекаем строки
	while ($row = mysqli_fetch_assoc($qr))
		$result[] = $row;

	// освобождаем ресурсы, связанные с хранением результата
	mysqli_free_result($qr);

	return $result;
}

/*
 * Извлекает из базы данных список пользователей
 */
function db_user_find_all($dbh)
{
	$query = 'SELECT * FROM users';
	$result = array();

	// выполняем запрос к базе данных
	$qr = mysqli_query($dbh, $query, MYSQLI_STORE_RESULT);
	if ($qr === false)
		db_handle_error($dbh);

	// последовательно извлекаем строки
	while ($row = mysqli_fetch_assoc($qr))
		$result[] = $row;

	// освобождаем ресурсы, связанные с хранением результата
	mysqli_free_result($qr);

	return $result;
}

/*
 * Выполняет поиск в базе данных и загрузку пользователя с указанным id
 */
function db_user_find_by_id($dbh, $id)
{
	$query = 'SELECT * FROM users WHERE id=?';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);

	mysqli_stmt_bind_param($stmt, 's', $id);
```

```
	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
	if ($qr === false)
		db_handle_error($dbh);

	// извлекаем результирующую строку
	$result = mysqli_fetch_assoc($qr);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	return $result;
}

/*
 * Выполняет загрузку всех SNO
 */
function db_sno($dbh)
{
	$query = 'SELECT * FROM SNO';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);


	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
	if ($qr === false)
		db_handle_error($dbh);

	// последовательно извлекаем строки
	while ($row = mysqli_fetch_assoc($qr))
		$result[] = $row;

	// освобождаем ресурсы, связанные с хранением результата
	mysqli_free_result($qr);

	return $result;
}

/*
 *  Выполняет загрузку всех OPF
 */
```

```
function db_opf($dbh)
{
      $query = 'SELECT * FROM OPF';

      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем результирующий набор строк
      $qr = mysqli_stmt_get_result($stmt);
      if ($qr === false)
            db_handle_error($dbh);

      // последовательно извлекаем строки
      while ($row = mysqli_fetch_assoc($qr))
            $result[] = $row;

      // освобождаем ресурсы, связанные с хранением результата
      mysqli_free_result($qr);

      return $result;
}


/*
 * Выполняет поиск в базе данных и возвращает id пользователя с указанным
email
 */
function db_user_id_find_by_email($dbh, $mail)
{
      $query = 'SELECT id  FROM users WHERE email=?';

      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);

      mysqli_stmt_bind_param($stmt, 's', $mail);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем результирующий набор строк
      $qr = mysqli_stmt_get_result($stmt);
      if ($qr === false)
            db_handle_error($dbh);

      // извлекаем результирующую строку
      $result = mysqli_fetch_assoc($qr);
```

```
	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	// извлекаем id
	$id_email=$result['id'];

	return $id_email;
}


/*
 * Выполняет поиск в базе данных и проверяет, есть ли пользователь с
указанным никнеймом
 */
function db_freedom_nick_find($dbh, $nickname)
{
	$query = 'SELECT COUNT(*)  FROM users WHERE nickname=?';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);

	mysqli_stmt_bind_param($stmt, 's', $nickname);

	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
	if ($qr === false)
		db_handle_error($dbh);

	// извлекаем результирующую строку
	$result = mysqli_fetch_assoc($qr);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	if ($result['COUNT(*)']==0)
	{
		return true;
	}

	return false;
}

/*
 * Выполняет поиск в базе данных и проверяет, есть ли пользователь
 * с указанным адресом электронной почты
 */
```

```
function db_freedom_email_find($dbh, $email)
{
	$query = 'SELECT COUNT(*)  FROM users WHERE email=?';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);

	mysqli_stmt_bind_param($stmt, 's', $email);

	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
	if ($qr === false)
		db_handle_error($dbh);

	// извлекаем результирующую строку
	$result = mysqli_fetch_assoc($qr);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	if ($result['COUNT(*)']==0)
	{
		return true;
	}

	return false;
}

/*
 * Выполняет проверяет, принадлежит ли компания пользователю
 */
function db_company_have_user($dbh, $id_company, $id_user)
{
	$query = 'SELECT COUNT(*)  FROM company WHERE Id=? and User_Id=?';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);

	mysqli_stmt_bind_param($stmt, 'ii', $id_company, $id_user);

	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
```

```
	if ($qr === false)
		db_handle_error($dbh);

	// извлекаем результирующую строку
	$result = mysqli_fetch_assoc($qr);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	if ($result['COUNT(*)']==0)
	{
		return false;
	}

	return true;
}

/*
 * Выполняет поиск в базе данных и загрузку пользователя с указанным
логином
 * (логином считаем адрес электронной почты и ник пользователя)
 */
function db_user_find_by_login($dbh, $login)
{
	$query = 'SELECT * FROM users WHERE email=? OR nickname=?';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);

	mysqli_stmt_bind_param($stmt, 'ss', $login, $login);

	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем результирующий набор строк
	$qr = mysqli_stmt_get_result($stmt);
	if ($qr === false)
		db_handle_error($dbh);

	// извлекаем результирующую строку
	$result = mysqli_fetch_assoc($qr);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	return $result;
}

/*
 *
```

```
 */
function db_socid_user_array($dbh, $id)
{
      $query = 'SELECT * FROM auth_social WHERE id_user=?';

      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);

      mysqli_stmt_bind_param($stmt, 'i', $id);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем результирующий набор строк
      $qr = mysqli_stmt_get_result($stmt);
      if ($qr === false)
            db_handle_error($dbh);

      // извлекаем результирующую строку
      $result = mysqli_fetch_assoc($qr);

      // освобождаем ресурсы, связанные с хранением результата и запроса
      mysqli_free_result($qr);
      mysqli_stmt_close($stmt);

      return $result;
}

/*
 * Выполняет поиск id пользователя по id социальной сети
 */
function db_id_find_by_socid($dbh, $soc, $socid)
{
      $query = 'SELECT id_user FROM `auth_social` WHERE '.$soc.'=?';
      //$query = 'SELECT id_user FROM `auth_social` WHERE vk=?';
      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);
      mysqli_stmt_bind_param($stmt, 'i',$socid);
      // выполняем запрос еи получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем результирующий набор строк
      $qr = mysqli_stmt_get_result($stmt);
      if ($qr === false)
            db_handle_error($dbh);

      // извлекаем результирующую строку
      $result = mysqli_fetch_assoc($qr);
```

```
	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_free_result($qr);
	mysqli_stmt_close($stmt);

	// извлекаем id
	$id_soc=$result['id_user'];
	return $id_soc;
}

/*
 * Выполняет удаление неактуальных данных из таблицы not_end_registration
 */
function db_del_not_reg($dbh, $hash)
{
	$query = 'DELETE FROM `not_end_registration` WHERE hash=?';
	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);
	mysqli_stmt_bind_param($stmt, 'i',$hash);
	// выполняем запрос еи получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// освобождаем ресурсы, связанные с хранением запроса
	mysqli_stmt_close($stmt);

	return true;
}

/*
 * Вставляет в базу данных строку с информацией о пользователе,
возвращает массив
 * с данными пользователя и его id в базе данных
 */
function db_user_insert($dbh, $user)
{
	$query = 'INSERT INTO
users(nickname,email,password,fullname,newsletter,status_active)
VALUES(?,?,?,?,?,?)';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);
	$user['status_active']=0;
	$pieces = explode("@", $user['email']);
	if($pieces[1]=='mail.ru')
	{
		$user['status_active']=1;
	}
	mysqli_stmt_bind_param($stmt, 'ssssii',
		$user['nickname'], $user['email'], $user['password'],
		$user['fullname'], $user['newsletter'],
$user['status_active']);
```

```
	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем идентификатор вставленной записи
	$user['id'] = mysqli_insert_id($dbh);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_stmt_close($stmt);

	return $user;
}

/*
 * Вставляет в базу данных временную информацию о смене пароля
 */
function forget_password_add($dbh, $email,$microtime)
{
	$query = 'REPLACE INTO forget_password(email_user,time)
VALUES(?,?)';

	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);
	$user['status_active']=0;
	mysqli_stmt_bind_param($stmt, 'ss', $email, $microtime);

	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// получаем идентификатор вставленной записи
	$user['id'] = mysqli_insert_id($dbh);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_stmt_close($stmt);

	return $user;
}

/*
 * Выполняет удаление неактуальных данных из таблицы not_end_registration
 */
function forget_password_del($dbh, $email)
{
	$query = 'DELETE FROM `forget_password` WHERE email_user=?';
	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);
	mysqli_stmt_bind_param($stmt, 's',$email);
	// выполняем запрос еи получаем результат
	if (mysqli_stmt_execute($stmt) === false)
```

```
            db_handle_error($dbh);

      // освобождаем ресурсы, связанные с хранением запроса
      mysqli_stmt_close($stmt);

      return true;
}


/*
 * Выполняет удаление компании из списка компаний
 */
function db_company_del($dbh, $id)
{
      $query = 'DELETE FROM `company` WHERE id=?';
      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);
      mysqli_stmt_bind_param($stmt, 'i',$id);
      // выполняем запрос еи получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // освобождаем ресурсы, связанные с хранением запроса
      mysqli_stmt_close($stmt);

      return true;
}

/*
 * Выполняет поиск данных из таблицы not_end_registration
 */
function forget_password_find_time($dbh, $email)
{
      $query = 'SELECT time FROM `forget_password` WHERE email_user=?';
      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);
      mysqli_stmt_bind_param($stmt, 's',$email);
      // выполняем запрос еи получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем результирующий набор строк
      $qr = mysqli_stmt_get_result($stmt);
      if ($qr === false)
            db_handle_error($dbh);

      // извлекаем результирующую строку
      $result = mysqli_fetch_assoc($qr);

      // освобождаем ресурсы, связанные с хранением результата и запроса
      mysqli_free_result($qr);
```

```
	mysqli_stmt_close($stmt);

	// извлекаем time
	$time=$result['time'];
	return $time;
}



/*
 * Вставляет в базу данных !ПУСТУЮ! строку с информацией о социальной
сети пользователя
 *
 */
function db_user_socid_insert_null($dbh, $db_user_id)
{

	$query = 'INSERT auth_social(id_user) VALUES(?)';


	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
		db_handle_error($dbh);
	$user['status_active']=0;
	mysqli_stmt_bind_param($stmt, 'i', $db_user_id);

	// выполняем запрос и получаем результат
	if (mysqli_stmt_execute($stmt) === false)
		db_handle_error($dbh);

	// освобождаем ресурсы, связанные с хранением результата и запроса
	mysqli_stmt_close($stmt);

	return true;
}



/*
 * Вставляет в базу данных строку с информацией о социальной сети
пользователя
 *
 */
function db_user_socid_insert($dbh, $soc_data, $db_user_id)
{
	$soc=$soc_data["social"];

	$query = 'INSERT INTO auth_social('.$soc.',id_user) VALUES(?,?) ON
DUPLICATE KEY UPDATE '.$soc.'=?';


	// подготовливаем запрос для выполнения
	$stmt = mysqli_prepare($dbh, $query);
	if ($stmt === false)
```

```
          db_handle_error($dbh);
     $user['status_active']=0;
     mysqli_stmt_bind_param($stmt, 'iii', $soc_data['id'], $db_user_id,
$soc_data['id']);

     // выполняем запрос и получаем результат
     if (mysqli_stmt_execute($stmt) === false)
          db_handle_error($dbh);

     // освобождаем ресурсы, связанные с хранением результата и запроса
     mysqli_stmt_close($stmt);

     return true;
}

/*
 * Вставляет в базу данных строку с информацией о id пользователя в соц
сети, с помощью которой он регистрируется
 */
function db_user_not_reg_insert($dbh, $soc, $socid)
{
     $query = 'INSERT IGNORE INTO not_end_registration(id,social,hash)
VALUES(?,?,?)';
     // подготовливаем запрос для выполнения
     $stmt = mysqli_prepare($dbh, $query);
     if ($stmt === false)
          db_handle_error($dbh);
     $user['status_active']=0;
     $hash=md5($socid.$soc.SALT2);
     mysqli_stmt_bind_param($stmt, 'iss',
          $socid, $soc, $hash);
     // выполняем запрос и получаем результат
     if (mysqli_stmt_execute($stmt) === false)
          db_handle_error($dbh);

     // получаем идентификатор вставленной записи

     // освобождаем ресурсы, связанные с хранением результата и запроса
     mysqli_stmt_close($stmt);

     return true;
}


/*
 * Вставляет в базу данных строку с информацией о id пользователя в соц
сети, с помощью которой он регистрируется
 */
function db_user_social_reg_ok_insert($dbh, $socid_soc)
{
     $query = 'SELECT id,social FROM not_end_registration WHERE hash=?';

     // подготовливаем запрос для выполнения
     $stmt = mysqli_prepare($dbh, $query);
     if ($stmt === false)
```

```
            db_handle_error($dbh);

      $hash=md5($socid_soc.SALT2);

      mysqli_stmt_bind_param($stmt, 's', $hash);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем результирующий набор строк
      $qr = mysqli_stmt_get_result($stmt);
      if ($qr === false)
            db_handle_error($dbh);

      // извлекаем результирующую строку
      $result = mysqli_fetch_assoc($qr);

      // освобождаем ресурсы, связанные с хранением результата и запроса
      mysqli_free_result($qr);
      mysqli_stmt_close($stmt);

      db_del_not_reg($dbh, $hash);

      return $result;
}


/*
 * Обновление данных пользователя в базе данных
 */
function db_user_update($dbh, $user)
{
      $query = 'UPDATE `users` SET
`nickname`=?,`email`=?,`password`=?,`fullname`=?,`newsletter`=? WHERE
`id`=?';
      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);
      mysqli_stmt_bind_param($stmt, 'ssssii',
            $user['nickname'], $user['email'], $user['password'],
            $user['fullname'], $user['newsletter'], $user['id']);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // получаем идентификатор вставленной записи
      $user['id'] = mysqli_insert_id($dbh);

      // освобождаем ресурсы, связанные с хранением результата и запроса
      mysqli_stmt_close($stmt);

      return $user;
```

```
}
/*
 * Обновление статуса пользователя в базе данных
 */
function db_user_status_update($dbh, $id_mail)
{
      $query = 'UPDATE `users` SET `status_active`=1 WHERE `id`=?';
      // подготовливаем запрос для выполнения
      $stmt = mysqli_prepare($dbh, $query);
      if ($stmt === false)
            db_handle_error($dbh);
      mysqli_stmt_bind_param($stmt, 'i', $id_mail);

      // выполняем запрос и получаем результат
      if (mysqli_stmt_execute($stmt) === false)
            db_handle_error($dbh);

      // освобождаем ресурсы, связанные с хранением результата и запроса
      mysqli_stmt_close($stmt);

      return true;
}
```

Отзыв на бакалаврскую работу

«Разработка веб-сервиса для расчета финансовых показателей»

студента 4 курса факультета математики и

информационных технологий

С.И. Абрамова

На экономическое состояние большинства организаций сильно влияют кредиты, которые при грамотном и рациональном использовании позволяют компаниям развиваться и получать конкурентные преимущества на рынке. При этом, с одной стороны, кредитные организации предлагают разнообразные формы и условия кредитования, из которых компании должны выделять оптимальные для себя, а с одной стороны сами банки вынуждены учитывать множество финансовых и нефинансовых характеристик для определения потенциальной платежеспособности компании-кредитора. Поэтому идея создания веб-сервисов для автоматизации процессов взаимодействия компаний с банками представляется весьма актуальной.

В данной бакалаврской работе была поставлена задача практической реализации части функционала подобного веб-сервиса с использованием современных технологий и инструментов быстрой разработки веб-приложений (открытые API социальных сетей, сценарии PHP, СУБД MySQL, JavaScript- и CSS-фреймворки).

В целом бакалаврская работа соответствует заявленной теме и раскрывает исследуемые технологии. В ходе выполнения выпускной работы С.И. Абрамов показал себя грамотным специалистом, способным самостоятельно формулировать и решать поставленные задачи.

Считаю, что бакалаврская работа С.И. Абрамова заслуживает оценки "отлично".

Руководитель бакалаврской работы
к.ф. – м.н., доцент кафедры
фундаментальной информатики А.В. Попов

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
МОРДОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.П. ОГАРЁВА"

ОТЗЫВ РЕЦЕНЗЕНТА
О ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Студента Абрамова Сергея Игоревича
(Фамилия, имя, отчество)

Факультет математики и информационных тех-ий
Кафедра Вычислительной информ. Группа 402
Направление подготовки Выч. инф. и инф. технологий
Квалификация (степень) бакалавр

Наименование темы: Разработка Веб-сервиса для расчета финансовых показателей

Рецензент Сухарев А.А., МГУ им. Н.П. Огарёва, каф. АиБ, к.ф.-м.н.
(Фамилия И.О., место работы, ученое звание, степень)

ОЦЕНКА ВЫПУСКНОЙ РАБОТЫ

№ п/п	Показатели		Оценка				
			5	4	3	2	0*
1.	Актуальность тематики работы		✓				
2.	Степень полноты обзора состояния вопроса и корректность постановки задачи			✓			
3.	Уровень и корректность использования в работе методов исследования, математического моделирования		✓				
4.	Степень комплексности работы, применение в ней знаний естественнонаучных, социально-экономических, общепрофессиональных и специальных дисциплин		✓				
5.	Ясность, четкость, последовательность и обоснованность изложения		✓				
6.	Применение современного математического и программного обеспечения, компьютерных технологий в работе		✓				
7.	Качество оформления (общий уровень грамотности, стиль изложения, качество иллюстраций, соответствие требованиям стандарта)		✓				
8.	Оригинальность и новизна полученных результатов (научных, конструкторских и технологических решений)		✓				
9.	Тип работы	фундаментальная с оригинальными результатами					
		реферативная					
		прикладная	✓				
10.	Рекомендации	к опубликованию					
		к внедрению	✓				
ИТОГОВАЯ ОЦЕНКА			5 (отлично)				

* – не оценивается (трудно оценить)

Отмеченные достоинства: Процесс проектирования и реализации веб-приложения описан очень подробно, что делает работу доступной для понимания.

Работа имеет практическое применение, разработанные авторами модули для регистрации и авторизации пользователей сайта с помощью нескольких социальных сетей, могут быть использованы в других веб-проектах.

Отмеченные недостатки: Недостаточно полно описан алгоритм определения кредитного лимита для организации, который вычисляется на основании введенных финансовых показателей.

Заключение: Указанный недостаток не умаляет достоинств работы, которая полностью удовлетворяет всем требованиям, предъявляемым к ВКР. Считаю, что выполненная работа заслуживает оценки „отлично", а её автор, Абрамов Артей Игоревич, заслуживает присвоения квалификации (степени) „бакалавр по направлению подготовки „Фундаментальная информатика и информационные технологии"".

15 июня 2018г. Рецензент Муф
(подпись)

Заявление о самостоятельном характере выполнения работы

Я, Абрамов Сергей Игоревич, обучающийся 4 курса направления подготовки 02.03.02 – Фундаментальная информатика и информационные технологии, заявляю, что в моей работе на тему «Разработка веб-сервиса для расчета финансовых показателей», представленной в Государственную экзаменационную комиссию для публичной защиты, не содержится элементов неправомерных заимствований.

Все прямые заимствования из печатных и электронных источников, а также ранее защищённых письменных работ, кандидатских и докторских диссертаций имеют соответствующие ссылки.

Я ознакомлен с действующим в Университете Положением о проверке работ обучающихся ФГБОУ ВО «МГУ им. Н.П. Огарёва» на наличие заимствований, в соответствии с которым обнаружение неправомерных заимствований является основанием для отрицательного отзыва руководителя работы.

13.06.2018

Подпись обучающегося Дата

Работа представлена для проверки в Системе «Антиплагиат.ВУЗ»

13.06.2018

Дата представления работы *подпись руководителя*

ОТЧЕТ
о результатах проверки работы обучающегося
на наличие заимствований

Ф.И.О. автора работы: Абрамов Сергей Игоревич

Тема работы: «Разработка веб-сервиса для расчета финансовых показателей»

Руководитель: Попов Андрей Владимирович

Представленная работа прошла проверку на наличие заимствований в системе «Антиплагиат.ВУЗ» (*или на иной аналогичной системе, анализа текстов на наличие заимствований, выбранной Университетом).*

Результаты автоматической проверки:

оригинальность 98,23%

цитирования 0,6%

заимствования 1,17%

Результаты анализа полного отчета на наличие заимствований:

правомерные заимствования: да, 1,17%

корректные цитирования: да, 0,6%

неправомерные заимствования: нет

признаки обхода системы: нет

Общее заключение об итоговой оригинальности работы и возможности ее допуска к защите:

Выпускная квалификационная работа Абрамова С. И. «Разработка веб-сервиса для расчета финансовых показателей» допускается к защите.

Руководитель работы — (подпись) — 15.06.2018 (дата) — А. В. Попов

канд. физ.-мат. наук

Заведующему кафедрой фундаментальной информатики
А. Г. Смольянову
студента 4 курса
очной формы обучения
(на бесплатной основе)
направления подготовки
02.03.02 – Фундаментальная информатика и информационные технологии
факультета математики и информационных технологий
ФГБОУ ВО «МГУ им. Н.П. Огарёва»
Абрамова Сергея Игоревича

заявление

Прошу разместить мою выпускную квалификационную работу на тему «Разработка веб-сервиса для расчета финансовых показателей» в электронной библиотечной системе университета в полном объеме.

20.06.2018
Дата

Подпись