**Содержание**

Введение 7

1 Теоретический раздел 8

1.1 Обзор предметной области 8

1.2 Способы защиты и подтверждения данных 9

1.3 Существующие решения 10

1.4 Постановка задачи 12

1.5 Выводы по главе 13

2 Проектирование приложения 14

2.1 Сущность «Пользователь» 14

2.2 Сущность «Ресурс» 15

2.3 Сущность «Данные» 15

2.3.1 Описание таблиц базы данных 16

2.4 Модуль генерации сильного пароля 16

2.5 Модуль шифрования и дешифрования AES 17

2.6 Модуль шифрования и дешифрования RSA 18

2.7 Графическое представление программы 18

2.7.1 Окно авторизации и регистрации пользователя 19

2.7.2 Страница ресурсов 20

2.7.3 Страница пользовательского профиля 22

3 Программное конструирование 24

3.1 Обоснование инструментальных средств разработки. 24

3.2 Описание структурного программного средства. 25

3.3 Вывод по главе 29

4 Тестирование программного средства 30

Заключение 34

Перечень использованных информационных ресурсов 35

Приложение А Техническое задание 36

Приложение Б Листинг программы 42

Введение

В современном цифровом мире, где каждый из нас активно использует множество онлайн-сервисов и приложений, управление и безопасность паролей становятся все более важными. Часто мы сталкиваемся с необходимостью запоминать и управлять множеством логинов и паролей, используемых для доступа к нашим аккаунтам. В таких условиях нередко возникает необходимость в надежном и удобном инструменте для хранения и управления паролями.

Целью работы является разработка приложения "Менеджер паролей", которое предоставит надежное решение для хранения парных значений Логин-Пароль. Приложение должно представлять из себя серверную и клиентскую часть, обладать графическим интерфейсом, обеспечивая интуитивно понятное и удобное взаимодействие с пользователем.

Одним из ключевых функциональных требований — возможность генерации сильного пароля. Это позволит пользователям создавать надежные пароли, устойчивые к взлому, и избавит их от необходимости придумывать сложные комбинации самостоятельно.

Однако безопасность данных является неотъемлемой частью приложения такого уровня. Пароли должны храниться в зашифрованном виде, а серверная часть приложения не должна работать с незашифрованными данными, чтобы обеспечить безопасность при взломе сервера.

Разработка такого приложения поможет пользователям эффективно организовать свои пароли, предотвратить их потерю или утечку, а также повысить безопасность своих онлайн-аккаунтов.

# 1 Теоретический раздел

В данном разделе приведены сведения о решаемой задаче и инструментах, с помощью которых будет разработано программное средство

## 1.1 Обзор предметной области

В наше время использование онлайн-сервисов и приложений стало неотъемлемой частью нашей повседневной жизни. При этом важным вопросом становится управление и безопасность паролей. Мы часто сталкиваемся с необходимостью запоминать и управлять множеством логинов и паролей для доступа к нашим аккаунтам. В связи с этим возникает потребность в надежном и удобном инструменте для хранения и управления паролями. Критериями выбора такого инструмента становятся:

* Надежность: Пользователи стремятся использовать инструменты, которые обеспечивают высокий уровень безопасности и защиты их паролей. Надежный менеджер паролей должен предоставлять механизмы шифрования и защиты данных, а также защиту от несанкционированного доступа.
* Удобство использования: Поскольку пользователи взаимодействуют с менеджером паролей на ежедневной основе, важно, чтобы приложение было интуитивно понятным и удобным в использовании. Это может включать простой интерфейс.
* Поддержка различных платформ: С учетом того, что пользователи часто используют различные устройства (компьютеры, смартфоны, планшеты), важно, чтобы менеджер паролей поддерживал несколько платформ и предлагал синхронизацию данных между ними.
* Генерация сильных паролей: Приложение должно предоставлять возможность генерировать сложные и уникальные пароли, чтобы усилить безопасность аккаунтов пользователей.

## 1.2 Способы защиты и подтверждения данных

В настоящее время существует несколько популярных способов защиты и подтверждения данных, например алгоритм симметричного шифрования AES и алгоритм асимметричного шифрования RSA, а также семейство криптографических алгоритмов SHA-2 — однонаправленных хеш-функций.

Рассмотрим алгоритмы шифрования AES и RSA. Симметричный алгоритм AES предполагает использование одного секрета (ключа) для шифрования и дешифрования информации, причем ключ задается пользователем и минимальная длина ключа составляет 16 байт. Асимметричный алгоритм RSA предполагает использование дух ключей: публичный и приватный. Публичный ключ используется только для шифрования данных, а приватный ключ используется для дешифрования данных, причем из публичного ключа невозможно получить приватный и наоборот. Ключи генерируются при помощи алгоритма, а не задаются пользователем вручную.

Однонаправленные хеш-функции, такие как алгоритмы SHA-2 (Secure Hash Algorithm 2), предназначены для создания хеш-сумм (отпечатков) сообщений произвольной длины. Хеш-функция принимает на вход сообщение и генерирует фиксированный размер хеш-значения. Одно из важных свойств хеш-функций - любое изменение входных данных приведет к изменению хеш-значения. Это позволяет использовать хеш-функции для подтверждения целостности данных.

Эти методы криптографической защиты и подтверждения данных широко используются в информационной безопасности для обеспечения конфиденциальности, целостности и аутентификации данных.

## 1.3 Существующие решения

Проведем анализ существующих информационных систем для хранения учетных данных: «Bitwarden», «1password».

На рисунке 1.1 представлен вид главной страницы приложения «Bitwarden». Данное приложение предназначено для хранения и просмотра пользовательских учетных данных, является бесплатным, кроссплатформенным, а в качестве технологии шифрования использует симметричный алгоритм AES.

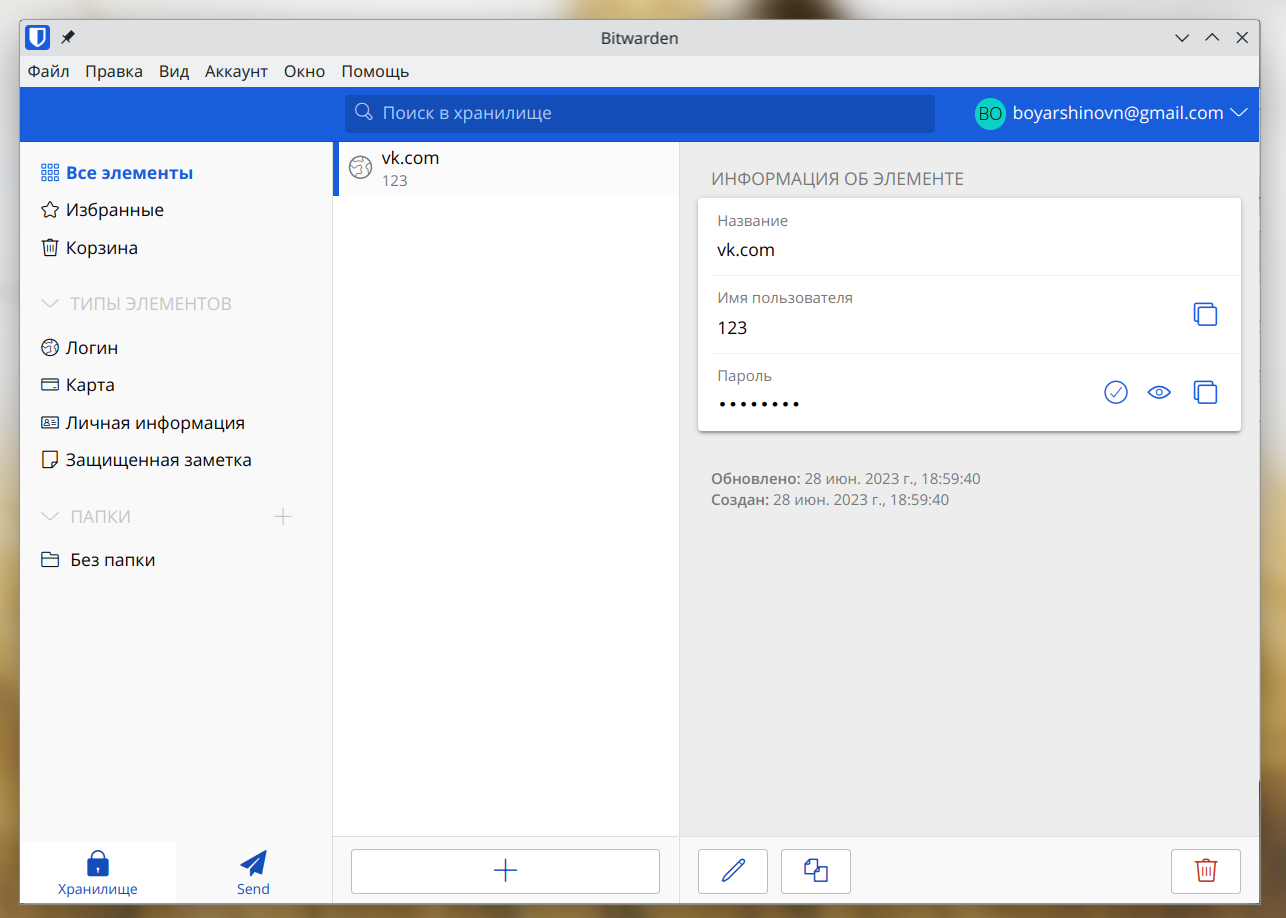


Рисунок 1.1 – Главная страница приложения «Bitwarden»

Интерфейс приложения является красивым, но не совсем удобным, из-за того, что при каждом входе в приложение нужно вводить пользовательских мастер-ключ, который будет использоваться как для шифрования, так для дешифрования сохраненных данных, в связи с чем приложение не совсем безопасно при использовании, так как постоянно хранит секрет в оперативной памяти.

На рисунке 1.2 представлен вид главной страницы приложения «1password». Данное приложение предназначено для хранения и просмотра пользовательских учетных данных, оно отличается тем, что использует отдельно ключ от аккаунта сервиса и мастер ключ для шифровки/дешифровки данных. Такой подход схож с «Bitwarden», но менее удобен, так как приходится постоянно хранить мастер ключ для использования приложения.

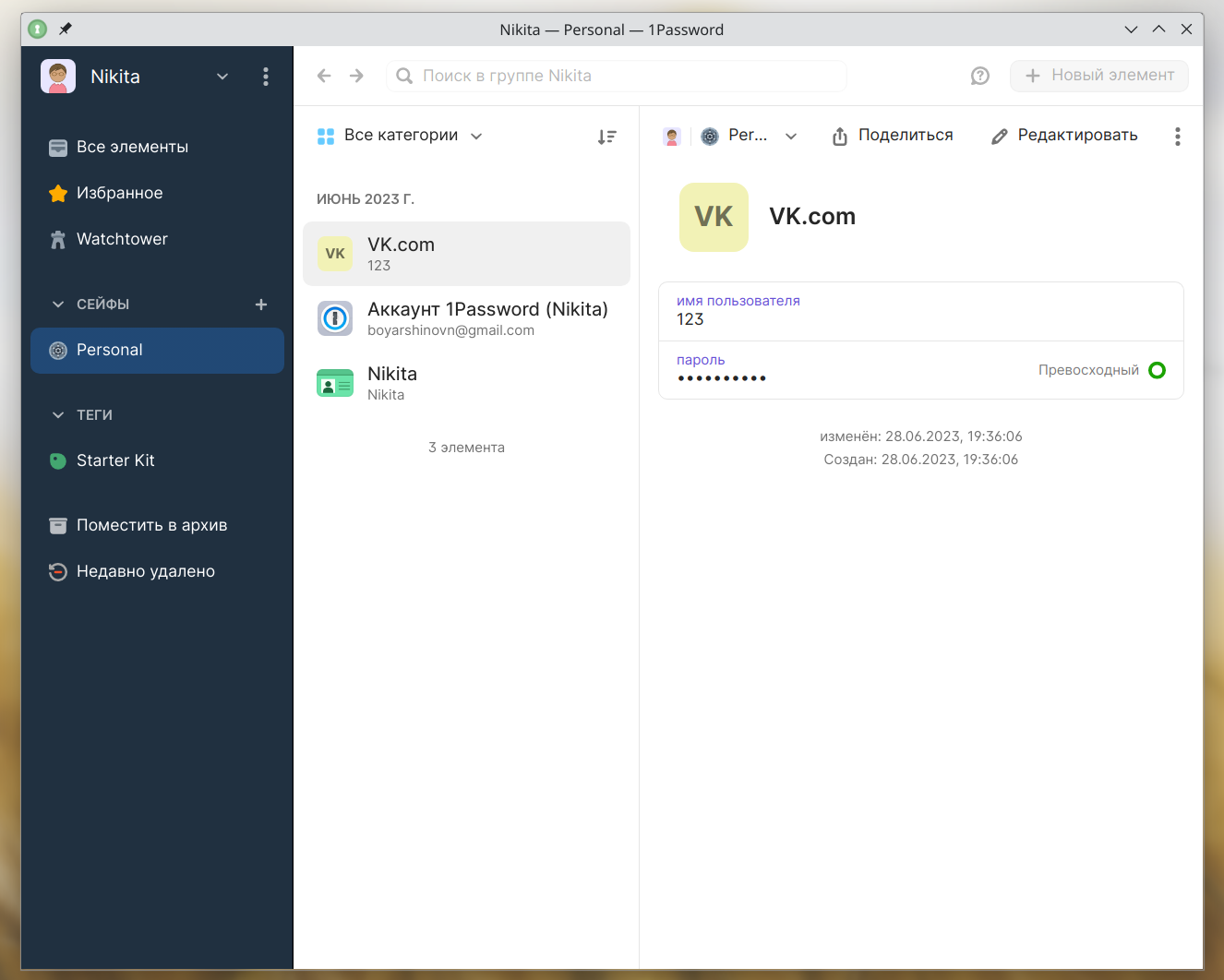


Рисунок 1.2 - Главная страница приложения «1password»

## 1.4 Постановка задачи

На основе проделанного предварительного анализа были сформулированы следующие принципы работы:

Принцип рационализма. Программный продукт должен иметь все основные инструменты для работы с данными пользователя, каждый из которых должен быть реализован в единственном числе.

Принцип устойчивости. Программный продукт создается для использования человеком, поэтому есть вероятность эксплуатационных ошибок. Приложение должно быть устойчиво к любым ошибочным действиям пользователя, для этого необходимо предусмотреть все варианты таких действий и обеспечить корректное реагирование программы на них.

Принцип доступности. Программный продукт предусматривает эксплуатацию пользователем, не имеющим специальных навыков. Приложение должно иметь систему подсказок пользователю и максимально простой, интуитивно-понятный графический интерфейс.

В результате работы была поставлена цель: разработать программное средство для реализации приложения "Менеджер паролей", позволяющее хранить пары Логин-Пароль. Был определен набор функциональных требований к разрабатываемому программному продукту, соответствующий нижеперечисленным принципам: 

* возможность сохранения пары логин и пароль;
* возможность чтения пары логин и пароль;
* система авторизации пользователя; 
* возможность группировать пользовательские пары логин и пароль;
* возможность удалять пары логин и пароль;
* возможность удалять созданные группы пар логин и пароль;
* корректное реагирование на ошибки пользователей; 
* система сообщений об ошибках и подсказках пользователю.

Для создания полноценного программного продукта, в процессе разработки необходимо выполнить все вышеперечисленные функциональные требования. Подробно требования к программному средству представлены в техническом задании (приложение А).

## 1.5 Выводы по главе

В данной главе была рассмотрена значимость разрабатываемого продукта, изучены существующие популярные способы защиты и подтверждения данных, а также изучены ошибки и недостатки существующих решений. Также были определены основные задачи и методы менеджера паролей. На основе проведенного обзора была поставлена задача и описаны функции, которые будет выполнять разрабатываемое программное средство.

# 2 Проектирование приложения

Проектирование приложения представляет собой многоступенчатый процесс его создания с использованием совокупности инструментальных средств. Основными этапами проектирования приложения является создание сущностей «Пользователь», «Ресурс» и «Данные», создание модулей функций генерации сильного пароля, шифрования, дешифрования как AES, так и RSA алгоритмов и их графическое отображение.

## 2.1 Сущность «Пользователь»

В разрабатываемом программном средстве сущность «Пользователь» играет важную как для шифрования, так и для дешифрования данных, а также помогает идентифицировать роль пользователя в системе. Обладает следующими полями:

* id – уникальный идентификатор пользователя в системе
* username – уникальное пользовательское имя
* hashed\_password – хеш пользовательского пароля
* role – пользовательская роль
* public\_key – публичный RSA ключ
* enc\_private\_key – закодированный (при помощи AES) приватный ключ RSA
* created\_at – дата создания
* updated\_at – дата обновления

Сущность умеет оперировать своими данными: выдавать значения полей, обновлять имя пользователя, обновлять пароль, обновлять ключи.

## 2.2 Сущность «Ресурс»

Данная сущность, представляет собой запись группы учетных данных пользователя, объединяет их под одним заголовком, позволяя группировать данные. Обладает следующими полями:

* id – уникальный идентификатор
* title – наименование ресурса (группы данных)
* owner\_id – идентификатор владельца ресурса
* created\_at – дата создания
* updated\_at – дата обновления

Сущность умеет оперировать своими данными: выдавать значения полей в виде пагинируемого списка, создавать, удалять и обновлять ресурс.

## 2.3 Сущность «Данные»

В разрабатываемом программном средстве сущность «Данные» является основным объектом, которых хранит пользовательские учетные записи. Обладает следующими полями:

* id – уникальный идентификатор
* username – имя пользователя
* enc\_password – зашифрованный пароль
* resource\_id – идентификатор ресурса (группы), к которому привязаны данные
* created\_at – дата создания
* updated\_at – дата обновления

Сущность умеет оперировать своими данными: выдавать значения полей в виде пагинируемого списка, создавать, удалять и обновлять данные.

### 2.3.1 Описание таблиц базы данных

В результате проектирования были созданы отношения (User, Resource, Datum), связи между которыми отображены на даталогической модели данных, которая представлена на рисунке 2.1.

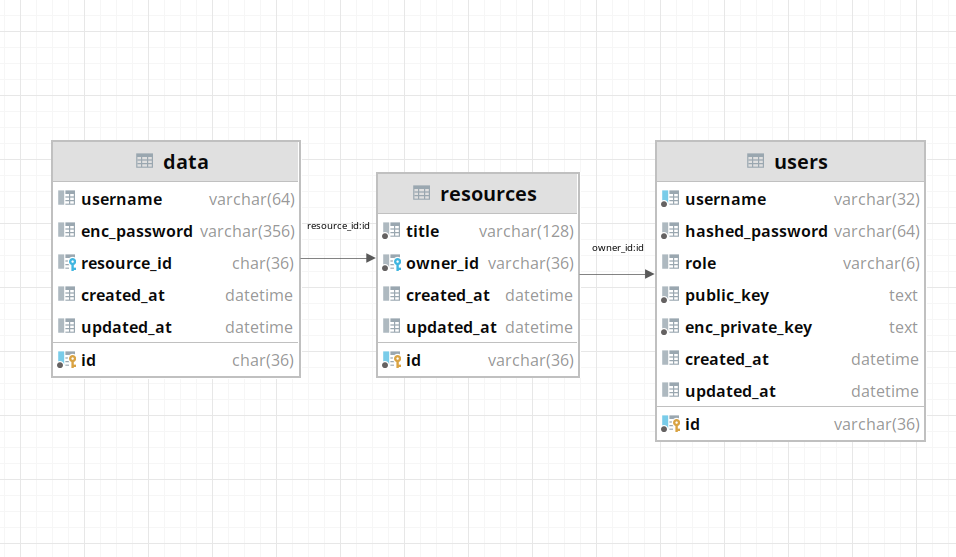


Рисунок 2.1 – Схема данных

Рассмотрим таблицы созданной даталогической модели. Модель «users» связана с моделью «resources» как один (user) ко многим (resources) по ключу «owner\_id». Сама модель «resources» связана с моделью «data» как один (resources) ко многим (data) по ключу «resource\_id». Схема является простой, логичной и удобной.

Основные программные модули приведены в Листинге Б.1.

### 2.4 Модуль генерации сильного пароля

В качестве удобного инструмента для пользователей был разработан модуль генерации сильного пароля. Данный модуль является тривиальным, генерирует пароль на стороне пользователя, что гарантирует безопасность, использует встроенные зависимости. Генерация происходит следующим образом: на первом шаге складываются ascii литералы в верхнем и нижнем регистре с цифрами и специальными символами. На следующем шаге: рандомным образом перебирается один из символов полученной строки и сохраняется в качестве выходного пароля. Это происходи столько раз, сколько требуется длина пароля. На выходе получается уникальный пароль, состоящий из символов ascii верхнего и нижнего регистров, цифр и специальных знаков.

### 2.5 Модуль шифрования и дешифрования AES

Модули шифрования и дешифрования AES являются одной из важнейшей функцией по защите данных.

Шифрование происходит следующим образом: функция шифрования принимает в качестве параметров: сообщение, необходимое зашифровать, секретный ключ, длиной от 16 до 32 байт. На следующем шаге создается шифр на основе секретного ключа и AES EAX mode - режима работы криптографических блочных шифров. После чего шифруется само сообщение на основе созданного шифра, полученное сообщение c одноразовым номером на основе шифра кодируется в base64 и декодируется в utf8. Полученная строка и является выходным данным.

Дешифрование происходит следующим образом: функция дешифровки на вход получает зашифрованное сообщение и секретный ключ. Из зашифрованного сообщения извлекается одноразовый номер на основе шифра и сам зашифрованный текст. Далее на основе секрета и одноразового номера восстанавливается шифр, использовавшийся при шифровании, после чего происходит дешифровка данных и декодирование в UTF-8.

Таким образом, модуль шифрования AES помогает быстро и безопасно шифровать и дешифровать информацию.

### 2.6 Модуль шифрования и дешифрования RSA

Модуль шифрования и дешифрования RSA является важным инструментом для защиты данных. Шифрование и дешифрование RSA осуществляется с помощью использования публичного и приватного ключей.

Шифрование происходит следующим образом: функция шифрования принимает в качестве параметров сообщение, которое необходимо зашифровать, и публичный ключ, который представляет собой строку в формате base64. Затем сообщение кодируется в UTF-8 и шифруется с использованием публичного ключа RSA с применением PKCS1\_OAEP (Optimal Asymmetric Encryption Padding). Полученное зашифрованное сообщение представляет собой строку, закодированную в формате base64.

Дешифрование происходит следующим образом: функция дешифрования принимает зашифрованное сообщение и приватный ключ, представленный в формате base64. Зашифрованное сообщение декодируется из формата base64, а затем расшифровывается с использованием приватного ключа RSA с использованием PKCS1\_OAEP. Расшифрованное сообщение преобразуется из байтового представления в строку UTF-8.

Таким образом, модуль шифрования и дешифрования RSA позволяет безопасно шифровать и дешифровать информацию с использованием публичного и приватного ключей. Это обеспечивает надежную защиту данных.

### 2.7 Графическое представление программы

В данном разделе описаны основные графические модули программного средства.

### 2.7.1 Окно авторизации и регистрации пользователя

Авторизация и регистрация пользователя представляет собой единое окно, в котором отображены поля ввода имени пользователя, пароля. Интерфейс авторизации и регистрации отображены на рисунках 2.2 и 2.3 соответственно.

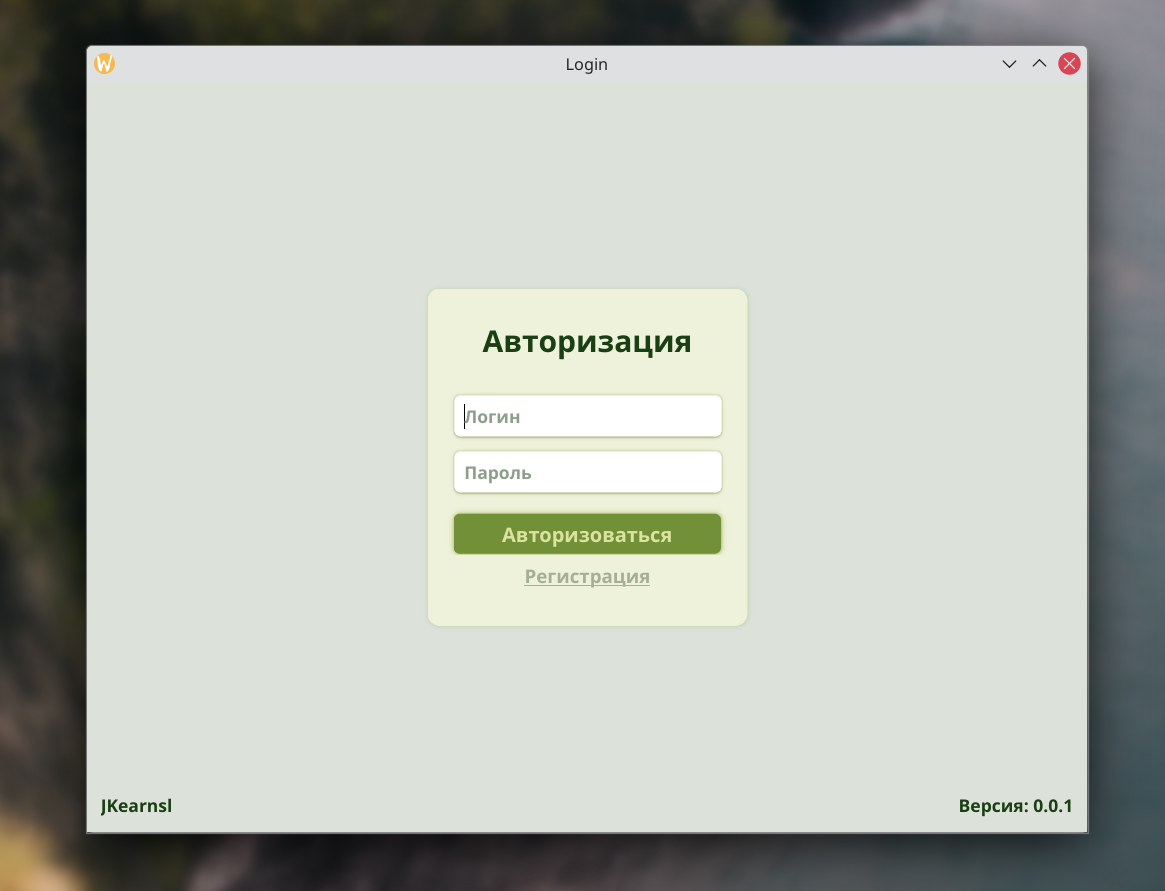


Рисунок 2.2 – Интерфейс авторизации

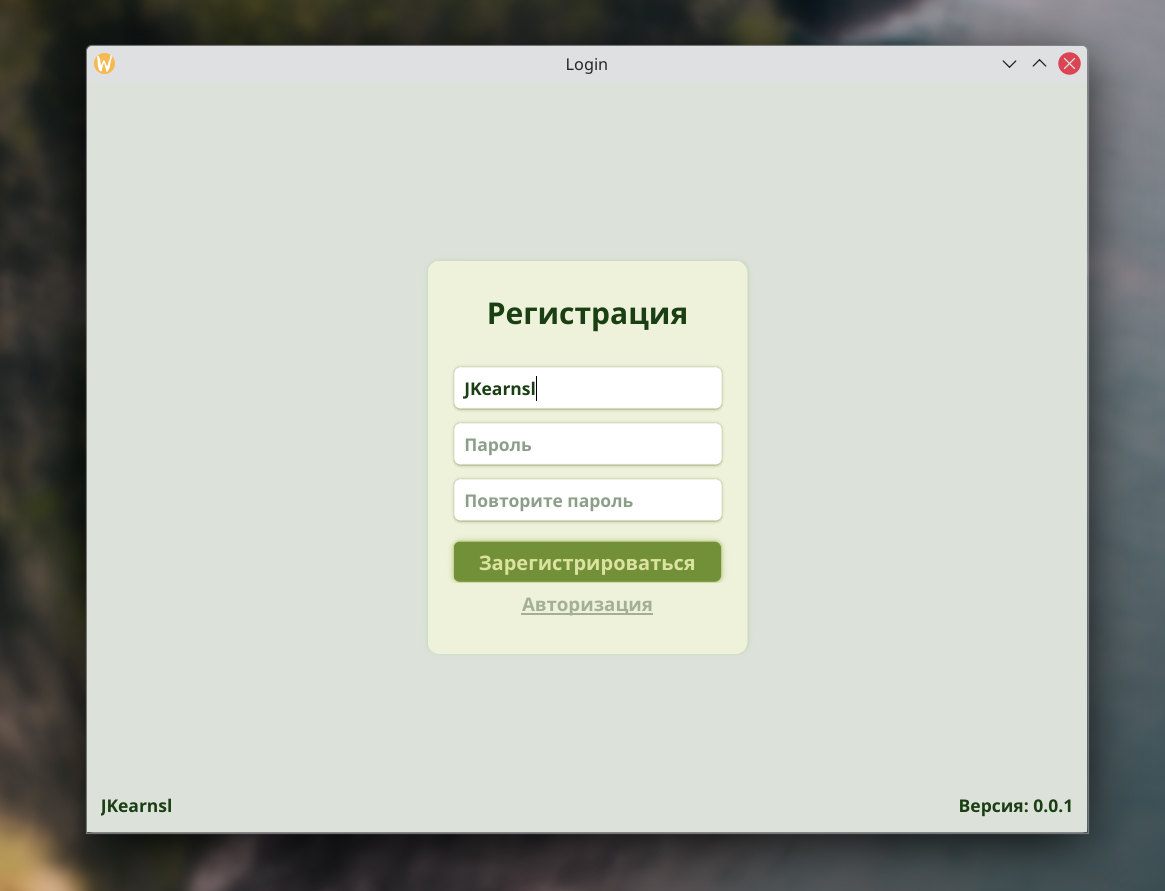


Рисунок 2.3 – Интерфейс регистрации

### 2.7.2 Страница ресурсов

Данная страница является дефакто главной в программе. Здесь отображен блок списка ресурсов, поисковая строка ввода, кнопки добавления ресурсов, кнопка добавления учетных данных, кнопки удаления и изменения. Подробно графический интерфейс отображен на рисунках 2.3, 2.5, 2.6 – соответственно.

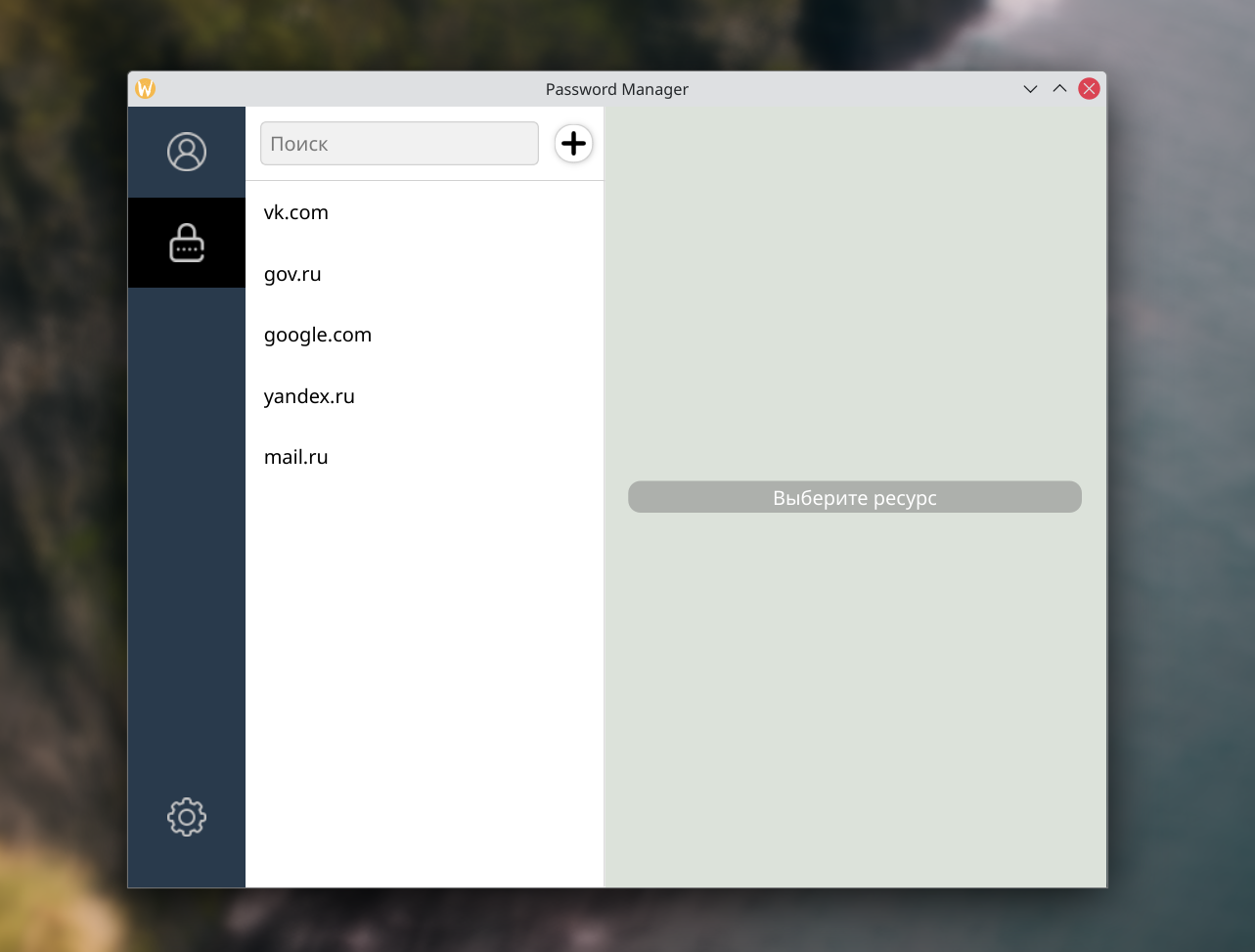


Рисунок 2.4 – Интерфейс страницы ресурсов

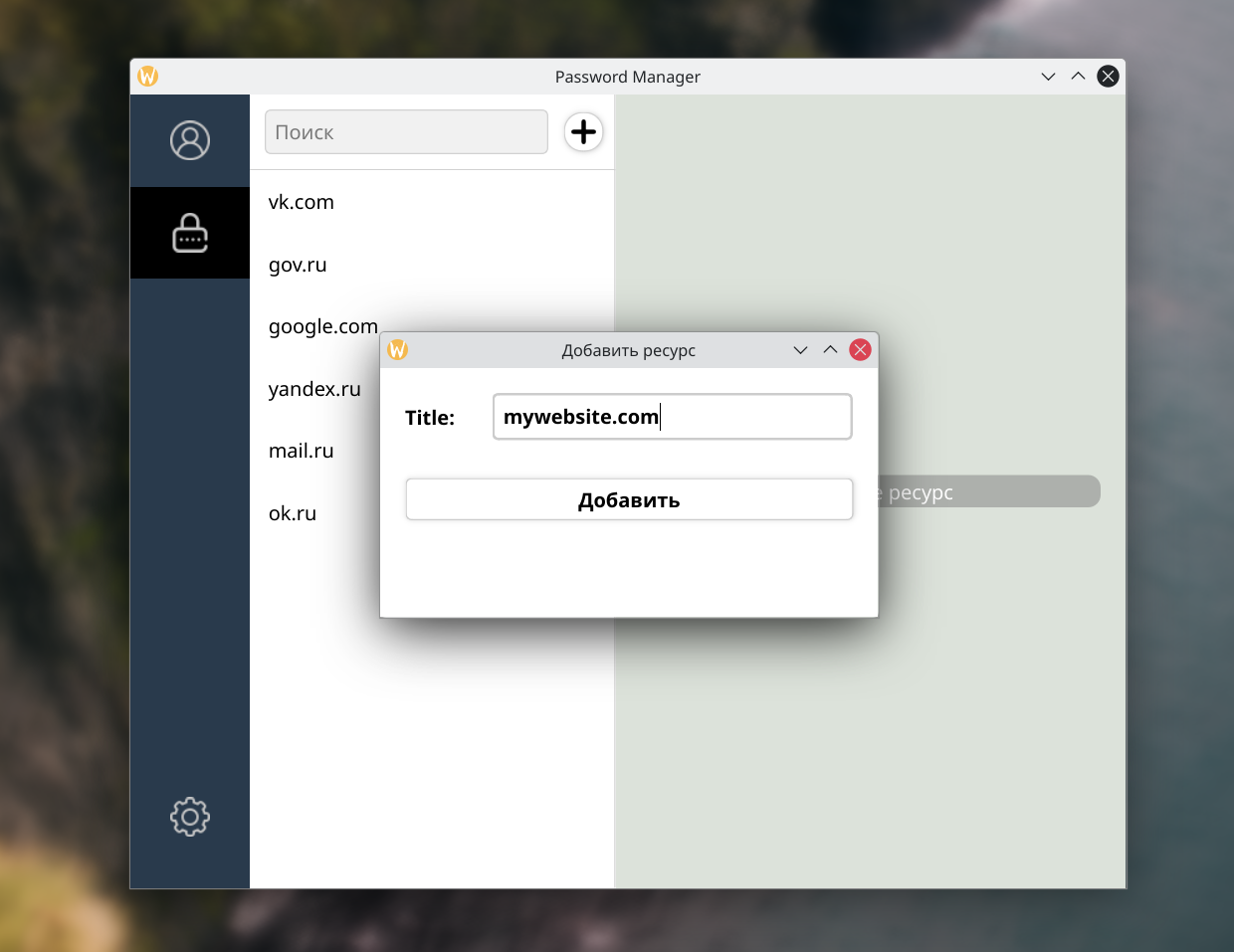


Рисунок 2.5 – Интерфейс добавления ресурса

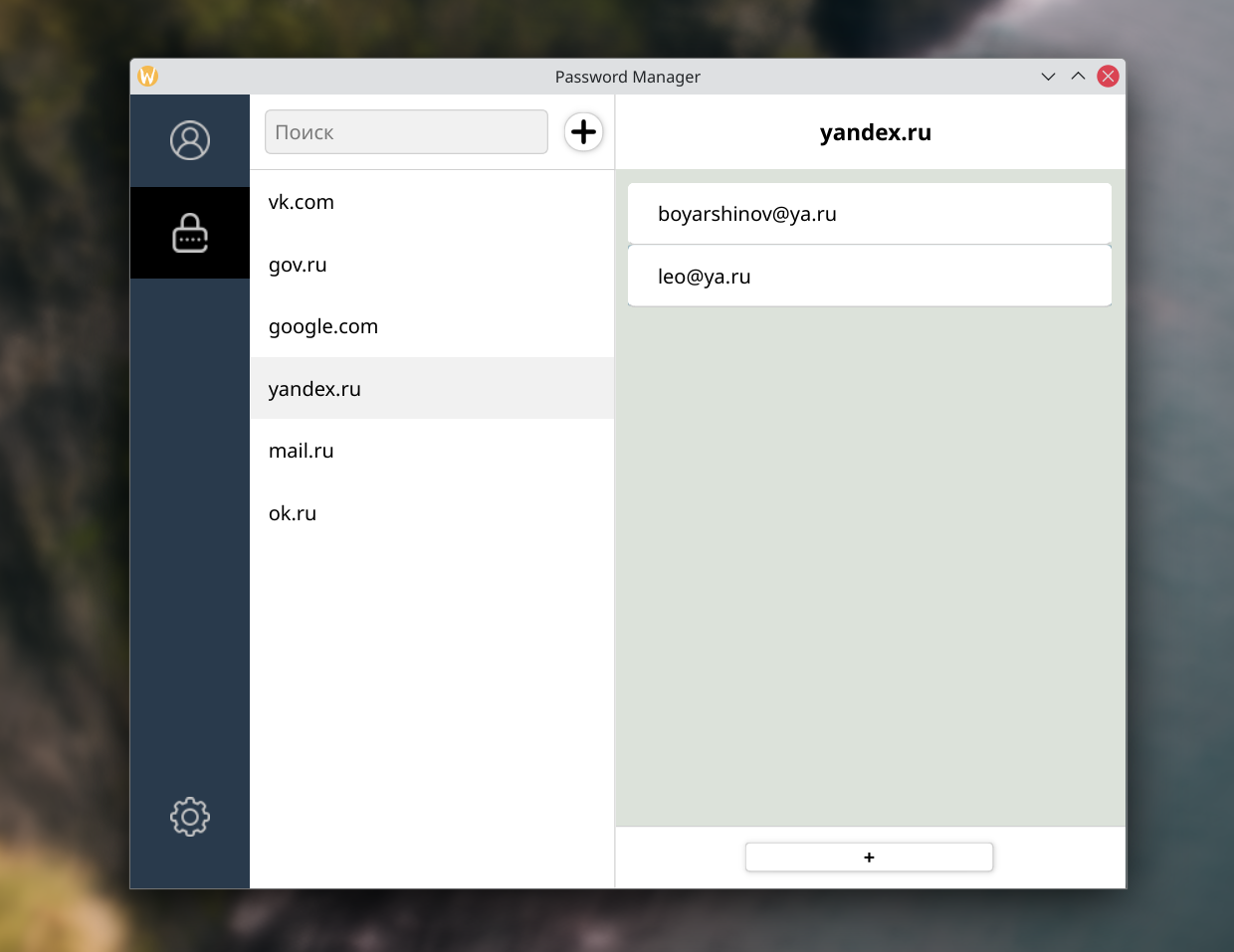


Рисунок 2.6 – Интерфейс списка данных

### 2.7.3 Страница пользовательского профиля

Страница пользовательского профиля содержит информацию о пользователе, а также позволяет редактировать её. На рисунке 2.7 подробно изображены отдельные кнопки для смены пользовательского пароля, а также смены ключей RSA и выхода из аккаунта.

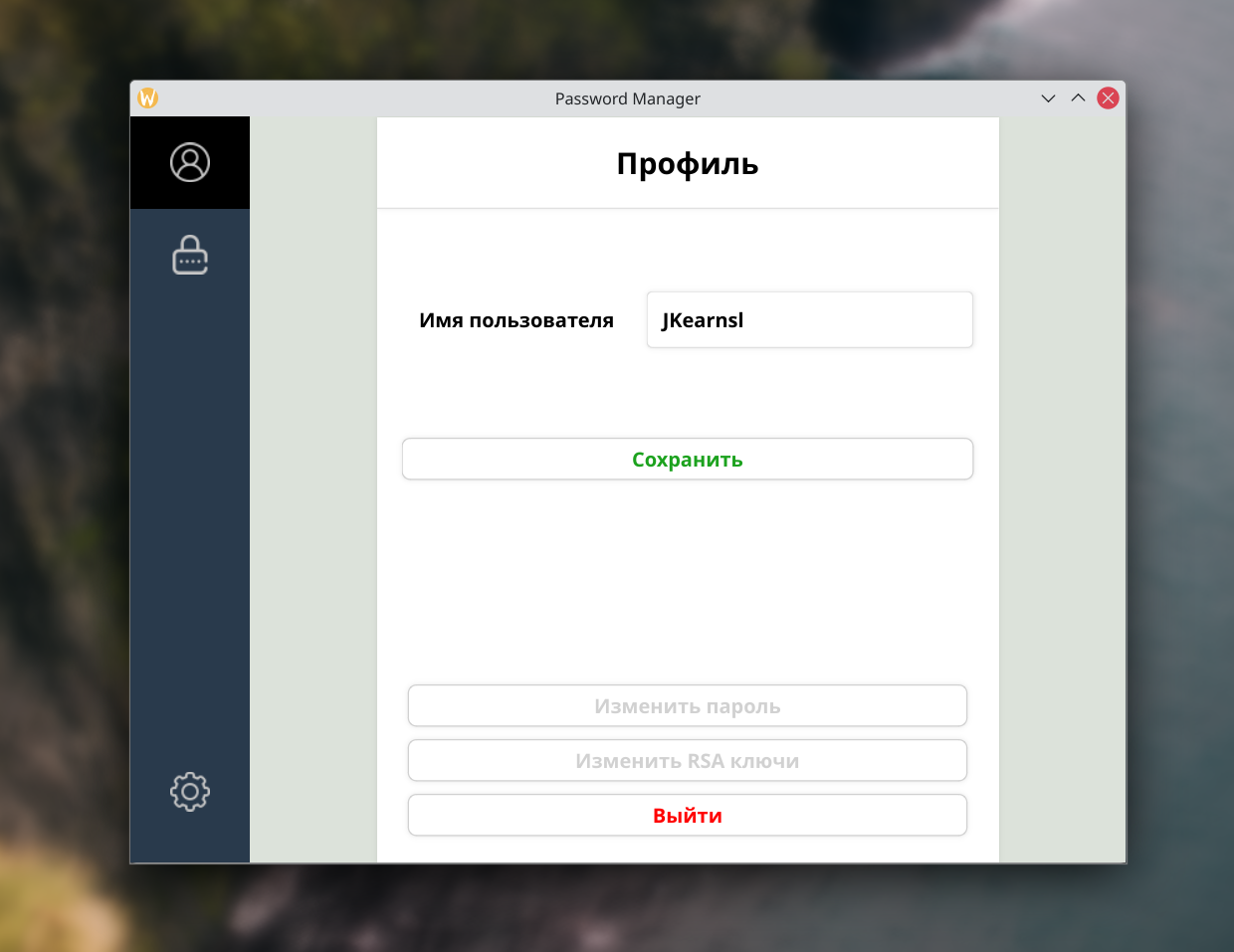


Рисунок 2.7 – Интерфейс списка данных

# 3 Программное конструирование

В данном разделе будут обоснованы выбор языка программирования, используемый для реализации программы, а также представлены основания выбора среды программирования. Будут определены и описаны основные подпрограммы разрабатываемого приложения.

## 3.1 Обоснование инструментальных средств разработки.

Для поставленной задачи создания клиент-серверного приложения существует множество языков программирования, библиотек и сред разработки. Однако, язык программирования Python, фреймворки PyQT, FastAPI и среда разработки PyCharm представляют собой мощный и удобный набор инструментов для решения данной задачи. Вот несколько причин, почему их стоит выбрать:

1. Простота и читаемость кода: Python является одним из наиболее простых и понятных языков программирования. Его синтаксис лаконичен и читаем, что упрощает разработку и понимание кода. Это особенно важно при работе с серверной частью приложения, где нужно обрабатывать и анализировать большое количество данных и иметь читаемую структуру.
2. Большое сообщество и поддержка: Python имеет огромное сообщество разработчиков, которое активно поддерживает язык и различные библиотеки. Это означает, что всегда можно найти помощь, документацию и примеры кода, что значительно упрощает процесс разработки и ускоряет решение возникающих проблем.
3. Фреймворк PyQT: QT - это популярный фреймворк для написания пользовательских приложений, которая имеет свою ветку для python, называемую PyQT. Она представляет удобное высокоуровневое API для создания окон, готовый набор виджетов и удобный набор типов для разработки кроссплатфоменных приложений.
4. Фреймворк FastAPI представляет собой удобный инструмент для разработки серверных приложений. Данное средство представляет набор функций для автоматической генерации документации.
5. Среда разработки PyCharm - это популярная интегрированная среда разработки (IDE) для Python, разработанная компанией JetBrains. Она предоставляет множество полезных функций, таких как автодополнение кода, отладка, инструменты для управления проектами и другие. PyCharm облегчает процесс разработки и повышает производительность разработчика.

В целом, выбор языка программирования Python, фреймворков PyQT, FastAPI и среды разработки PyCharm обеспечивает эффективность, простоту и удобство в разработке. Эти инструменты позволяют разработчикам быстро и легко реализовывать функциональность программы, обеспечивая при этом читаемость кода и хорошую поддержку со стороны сообщества.

## 3.2 Описание структурного программного средства.

Для работы программного средства реализованы следующие функции и методы, которые описаны в таблицах 3.1 и 3.2

Таблица 3.1 – Описание реализованных функций для работы серверной части приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Описание функций | Возвращаемое значение |
| 1 | 2 | 3 | 4 |
| get\_me | repository, current\_user | Функция получения модели текущего пользователя | User |
| get\_keys | repository, current\_user | Функция получения пары RSA ключей | Keys |
| update\_username | username,  new\_hpassword,  old\_hpassword,  current\_user | Функция для обновления имени пользователя | - |
| update\_password | new\_hpassword,  old\_hpassword,  new\_encPrivatekey,  current\_user | Функция для обновления пароля пользователя | - |
| update\_keys | public\_key  enc\_private\_key  hashed\_password,  current\_user | Функция для обновления пар RSA ключей | - |
| delete\_me | repository,  current\_user | Функция для удаления пользователя | - |
| get\_resource\_list | page,  per\_page,  query,  order\_by,  current\_user | Функция для получения списка ресурсов | Resources |
| create\_resource | title  current\_user | Функция для создания ресурса | Resource |
| delete\_resource | message  current\_user | Функция для удаления ресурса | - |
| update\_resource | resource\_id,  title,  current\_user | Функция для обновления ресурса | - |
| create\_datum | resource\_id  current\_user | Функция для создания единицы учетных данных | Datum |
| get\_data\_list | page,  per\_page,  query,  order\_by,  current\_user | Функция для получения списка учетных данных | Data |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| delete\_datum | datum\_id,  current\_user | Функция для удаления единицы учетных данных | - |
| update\_datum | datum\_id,  username,  enc\_password,  current\_user | Функция для обновления единицы учетных данных | - |
| reg\_root\_api\_router | is\_debug | Функция регистрирует маршруты для прослушивания | APIRouter |

Таблица 3.2 – Описание реализованных функций для работы клиентской части приложения

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Описание функций | Возвращаемое значение |
| 1 | 2 | 3 | 4 |
| is\_auth | api\_service | Функция для получения статуса авторизации пользователя | bool |
| get\_http\_session | - | Функция чтения и инициализации http сессии | Client |
| switch\_auth\_state | auth\_state | Функция для смены состояния формы авторизации | - |
| signin | username,  password | Функция для авторизации пользователя | bool |
| signup | username, password, repeat\_password | Функция для регистрации пользователя | bool |
| save\_session | - | Функция для сохранения текущей сессии | - |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| logout | api\_service | Функция для удаление пользовательской сессии | - |
| search\_resources | query | Функция для поиска ресурсов по запросу | - |
| load\_resources | page,  per\_page,  query,  order\_by | Функция для загрузки списка ресурсов | - |
| add\_resource | title | Функция для добавления ресурса | - |
| delete\_resource | resource\_id | Функция для удаления ресурса | - |
| update\_resource | resource\_id,  title | Функция для обновления ресурса | - |
| load\_data\_list | page,  per\_page,  resource\_id | Функция для загрузки списка учетных данных | - |
| add\_datum | resource\_id,  username,  password | Функция для добавления учетных данных | - |
| decrypt\_datum\_password | user\_password | Функция для расшифровки пароля пользователя | - |
| generate\_keys | bites | Функция генерации ключей RSA | private\_key, public\_key |
| encrypt\_aes | message,  secret | Функция шифрования AES | enc\_message |

3.3 Вывод по главе

В данной главе был обоснован выбор языка программирования, а также

среда программирования для создаваемого программного средства. Кроме того, были описаны основные подпрограммы программного средства. В таблице было приведено описание каждой из подпрограмм с указанием типа входных параметров и возвращаемого значения.

4 Тестирование программного средства

В данном разделе будет описано и протестирована работа программного приложения и рассмотрены примеры с подробными входными данными и подтверждающими работу программного средства рисунками.

4.1 Описание контрольных примеров

Для тестирования разработанного программного средства было создано несколько контрольных примеров, позволяющих проверить эффективность и корректность работы при различных условиях.

4.2 Описание результатов тестирования

В примере показана работа основного функционала, а именно страница авторизации и вывод главной страницы приложения. Страница авторизации представляет собой форму ввода данных, кнопки авторизации и переключения между авторизацией и регистрацией (рисунок 4.1)

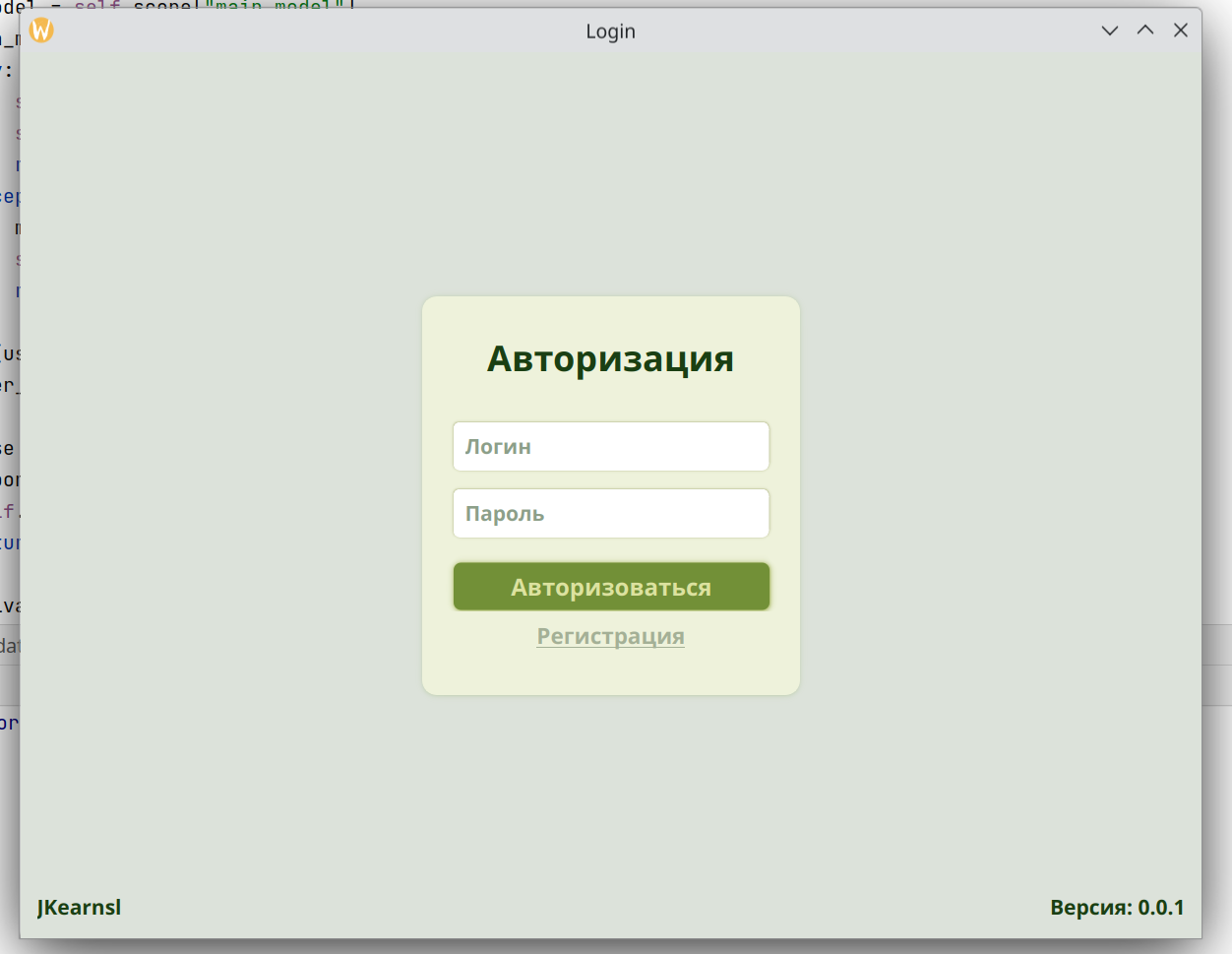


Рисунок 4.1 – Интерфейс страницы авторизации

Если пользователь введет логин и пароль и нажмет на кнопку «Авторизоваться», то произойдет авторизация пользователя и откроется главная страница (рисунок 4.2).

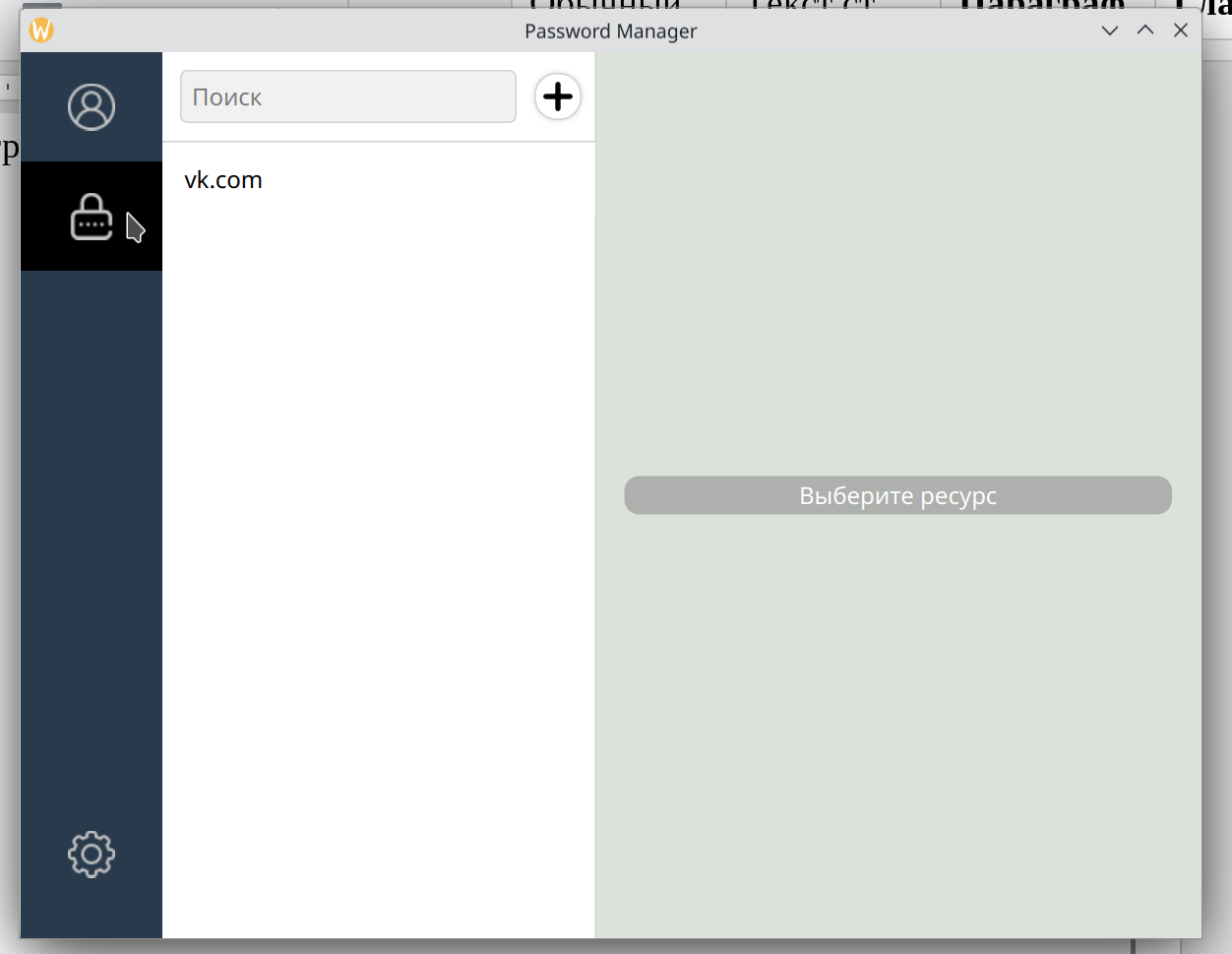


Рисунок 4.2 – Интерфейс главной страницы

На данной странице отображен список ресурсов, строка поиска, кнопка добавления нового ресурса. При нажатии на существующий ресурс «vk.com» в крайнем правом блоке отображен список данных выбранного ресурса (рисунок 4.3).

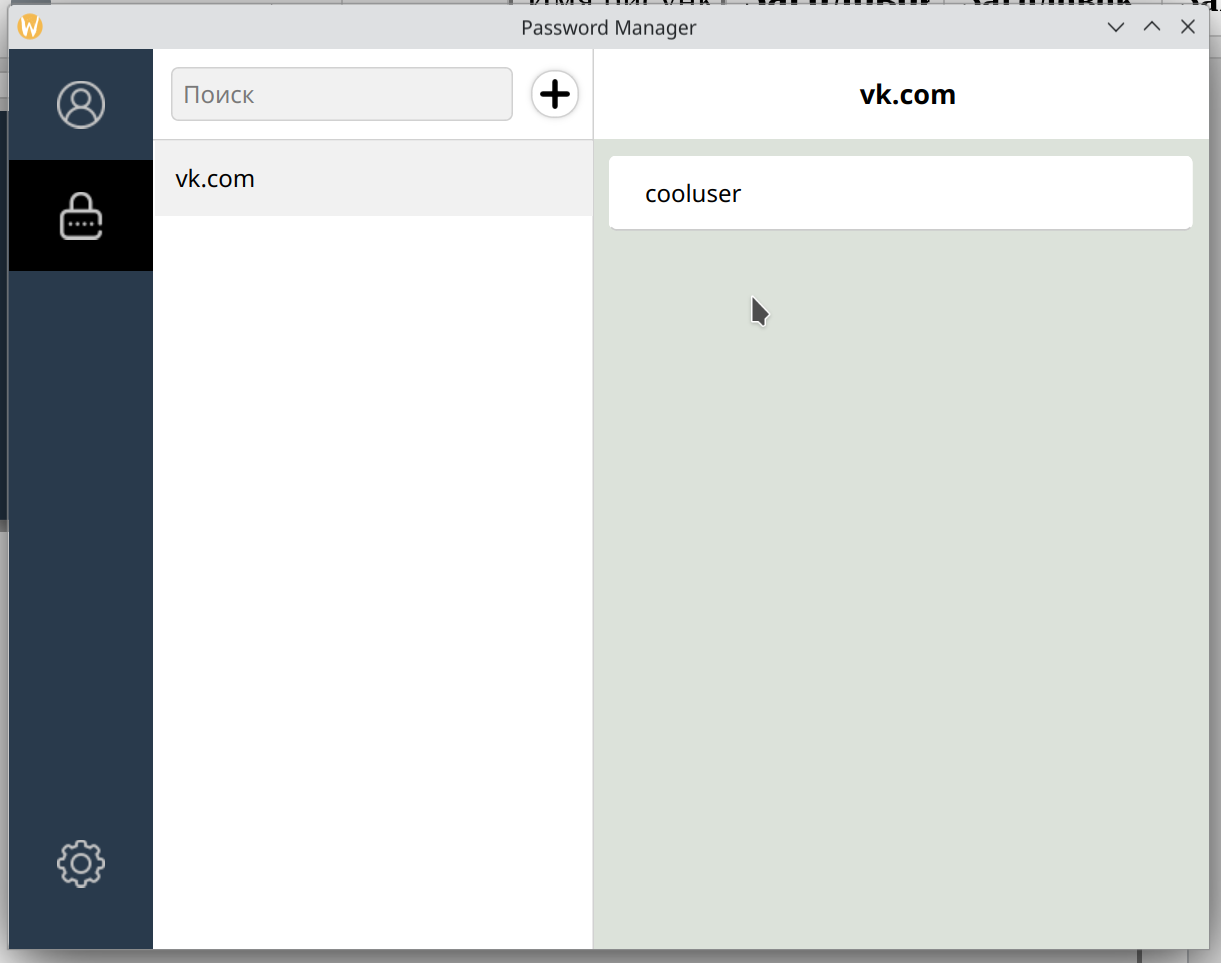


Рисунок 4.3 – Список данных

При нажатии на ячейку данных, отображается её содержимое (рисунок 4.4)

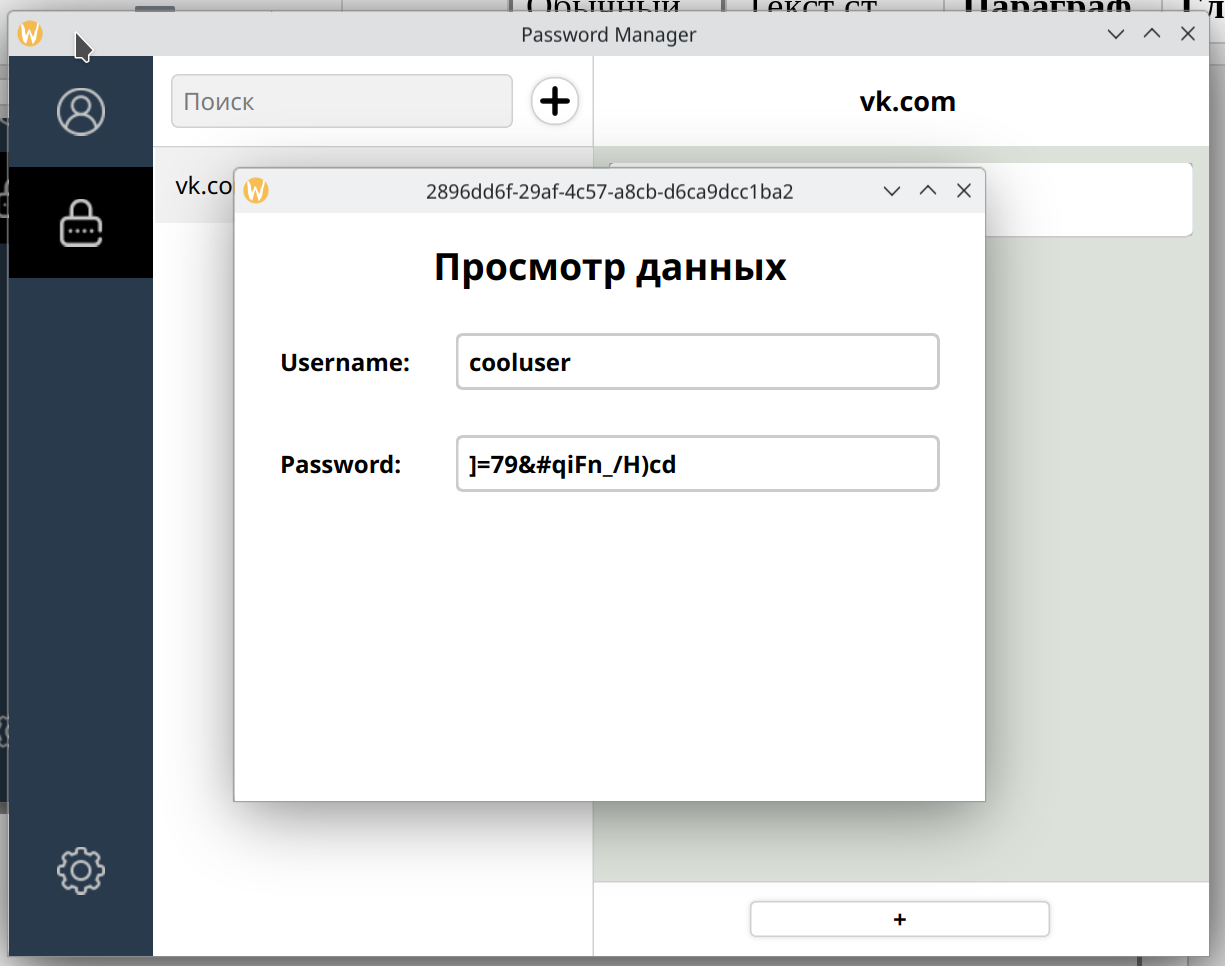


Рисунок 4.4 – Просмотр данных

4.3 Вывод по главе

В данной главе была показана работа программного средства. Программное средство реализует все функции, которые были заданы в техническом задании. Также была проведена проверка работы программного средства, которое показало устойчивое поведение и стабильную работу.

Заключение

В ходе выполнения выпускной квалификационной работы были решены следующие задачи.

Проведен обзор предметной области и выполнен сравнительный анализ информационных систем для аренды автомобилей.

Выполнено проектирование инфологической и даталогической моделей информационной системы.

Обоснован выбор программного инструментария: серверная часть система реализована на базе Python + FastAPI, клиентская часть системы реализована посредством Python + QT.

Проведено программное конструирование информационной системы и разработана система, отвечающая всем требованиям технического задания.

Перечень использованных информационных ресурсов

1. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ = Introduction to Algorithms / Под ред. И. В. Красикова. — 2-е изд. — М.: Вильямс, 2005. — 1296 с. — ISBN 5-8459-0857-4
2. "Advanced Encryption Standard (AES)" (National Institute of Standards and Technology) [Электронный ресурс] URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf
3. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. — 2nd Edition. — Wiley, 1996. — 758 p. — ISBN 978-0471128458
4. Stallings, W. Cryptography and Network Security: Principles and Practice. — 7th Edition. — Pearson, 2016. — 752 p. — ISBN 978-0134444284
5. "Cryptography: An Introduction" (Nigel Smart) [Электронный ресурс] URL: https://www.cs.bris.ac.uk/~nigel/Crypto\_Book/

Приложение А Техническое задание

|  |  |
| --- | --- |
|  | УТВЕРЖДЕНО  Зав. кафедрой «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_ В.В. Долгов  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

## А.1. Общие сведения

### А.1.1 Имя программы

«PasswordManager»

### А.1.2 Полное наименование системы

Клиент-серверное приложение для хранения паролей «PasswordManager»

### А.1.3 Перечень документов, на основании которых создается система

Основанием для разработки является задание к практической работе, согласованное с руководителем практики компании ФГАНУ НИИ «Спецвузавтоматика» Гуфан К.Ю. с одной стороны, студентом гр. ВПР-22 Бояршиновым Н.О., именуемым в дальнейшем исполнителем с другой стороны, утвержденному заведующим кафедрой «ПОВТиАС» Долговым Василием Валерьевичем.

### А.1.4 Порядок оформления и предъявления приложения

Приложение оформляется в виде раздельных блоков программной реализации, которые, в ходе выполнения, могут пересекаться с друг другом. Предъявляется приложение в виде готовой системы с возможностью ввода входных данных пользователем.

## А.2 Назначение приложения

Основным назначением приложения является безопасное хранение пар «Логин-Пароль» пользователей.

## А.3.Требования к информационной системе

### А.3.1 Требования к системе в целом

#### А.3.1.1 Требования к графическому дизайну приложения

Дизайн должен иллюстрировать результирующие данные в ходе работы программы. Пользователь должен иметь возможность выбрать пункты меню, легко создавать группы пользовательских данных и сами данны, а также удалять, редактировать и просматривать их. Дизайн приложения должен быть минималистичен, цвета должны быть приятны глазу и сочетаться между собой.

#### А.3.1.2 Требования к контенту и наполнению приложения

В приложении должно быть основное меню навигации, в котором пользователь ориентируется с помощью кнопок. На кнопках должны быть изображены иконки, символизирующие основные названия разделов (страниц) приложения. При переходе между разделами, состояния предыдущих разделов приложения не должны сбрасываться.

#### А.3.1.3 Требования к окнам приложения

Компоновка окон приложения должна обеспечивать минимальный объём действий со стороны пользователя. Приложение должно отзывчиво ко всем инструментам интерфейса и обеспечивать пользователя понятным и приятным дизайном.

#### А.3.1.6 Язык приложения

Русский.

### А.3.2 Требования к функциям (возможностям), выполняемых приложением

#### А.3.2.1 Основные требования

Приложение должно представлять собой графическое взаимодействие с пользователем для удобной возможности управления учетной записью пользователя, а также безопасное сохранения пользовательских данных.

## A.4 Требования к программе или программному изделию

### А.4.1 Требования к функциональным характеристикам

Программное средство должно осуществлять следующие функции:

* осуществлять взаимодействие с сервером с целью сохранения данных;
* регистрировать пользователя в системе и предоставлять ему управление своим профилем;
* шифровать и дешифровать пользовательские данные;
* предоставлять функции генерации сильного пароля;
* просматривать и изменять ресурсы и сохраненные данные.

### А.4.2 Требования к надежности

Надежное функционирование программы должно быть обеспечено выполнением совокупности нижеописанных мероприятий:

* выполнение требований ГОСТ 51188-98;
* защита информации;
* организацией бесперебойного питания технических средств;
* использованием лицензионного программного обеспечения.

### А.4.3 Условия эксплуатации

Для функционирования программного продукта необходимо соблюдение всех требований и правил эксплуатации устройства.

Высокая квалификация пользователя программного средства не требуется. Дополнительных требований и ограничений не вводится.

Требования к персоналу, работающему с данным программным продуктом – общие знания вычислительной техники.

### А.4.4 Требования к составу и параметрам технических средств

Состав технических средств на стороне клиента: устройство с операционной системой.

### А.4.5 Требования к информационной и программной совместимости

Для функционирования программного средства на стороне клиента необходимо следующее программное обеспечение: CPython 3.11.

#### А.4.5.1 Язык программирования

Для разработки программного средства должен быть использован язык программирования Python.

#### А.4.5.2 Операционная система

Кроссплатформенное ПО

### А.4.6 Требования к упаковке и маркировке

Требования к упаковке и маркировке программного средства не предъявляется.

## А.5 Порядок контроля и приемки

Порядок и контроль приёмки определяются заведующим кафедрой «ПОВТиАС» и основаны на демонстрации знаний технологии и умении создавать программные средства для различных предметных областей.

Главным требованием к приемке является наличие правильно работающего программного средства с тестовым примером и отчета, представленного в печатном виде.

Разработчик технического задания /Бояршинов Н.О. /

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

Приложение Б Листинг программы

Листинг Б.1 – Основная активность приложения

import logging

import redis.asyncio as redis

from fastapi import FastAPI

from fastapi.exceptions import RequestValidationError

from src.models import tables

from src.db import create\_sqlite\_async\_session

from src.middleware import JWTMiddlewareHTTP

from src.config import load\_ini\_config

from src.exceptions import APIError, handle\_api\_error, handle\_404\_error, handle\_pydantic\_error

from src.router import reg\_root\_api\_router

from src.utils import RedisClient

from src.utils.fakeredis import FakeRedisPool

from src.utils.openapi import custom\_openapi

config = load\_ini\_config('./config.ini')

log = logging.getLogger(\_\_name\_\_)

log.debug("Инициализация приложения FastAPI.")

app = FastAPI(

title=config.BASE.TITLE,

debug=config.DEBUG,

version=config.BASE.VERSION,

description=config.BASE.DESCRIPTION,

root\_path="/api/v1" if not config.DEBUG else "",

docs\_url="/api/docs" if config.DEBUG else "/docs",

redoc\_url="/api/redoc" if config.DEBUG else "/redoc",

contact={

"name": config.BASE.CONTACT.NAME,

"url": config.BASE.CONTACT.URL,

"email": config.BASE.CONTACT.EMAIL,

}

)

async def init\_sqlite\_db():

engine, session = create\_sqlite\_async\_session(

database='database.db',

echo=config.DEBUG,

)

app.state.db\_session = session

async with engine.begin() as conn:

# await conn.run\_sync(tables.Base.metadata.drop\_all)

await conn.run\_sync(tables.Base.metadata.create\_all)

async def redis\_pool(db: int = 0):

return await redis.from\_url(

f"redis://:{config.DB.REDIS.PASSWORD}@{config.DB.REDIS.HOST}:{config.DB.REDIS.PORT}/{db}",

encoding="utf-8",

decode\_responses=True,

)

@app.on\_event("startup")

async def on\_startup():

log.debug("Выполнение FastAPI startup event handler.")

await init\_sqlite\_db()

if config.DB.REDIS:

rp = await redis\_pool()

else:

rp = FakeRedisPool()

app.state.redis = RedisClient(rp)

@app.on\_event("shutdown")

async def on\_shutdown():

log.debug("Выполнение FastAPI shutdown event handler.")

await app.state.redis.close()

app.openapi = lambda: custom\_openapi(app)

app.state.config = config

log.debug("Добавление маршрутов")

app.include\_router(reg\_root\_api\_router(config.DEBUG))

log.debug("Регистрация обработчиков исключений.")

app.add\_exception\_handler(APIError, handle\_api\_error)

app.add\_exception\_handler(404, handle\_404\_error)

app.add\_exception\_handler(RequestValidationError, handle\_pydantic\_error)

log.debug("Регистрация middleware.")

app.add\_middleware(JWTMiddlewareHTTP)