«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.01 – Информатика и вычислительная техника
Профиль	Системы автоматизированного проектирования
Факультет	Компьютерных технологий и информатики
Кафедра	систем автоматизированного проектирования

К защите допустить
Зав. кафедрой, к.т.н., доцент _____ Бутусов Д.Н.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ТЕМА: Разработка серверной части приложения для хранения паролей

Студент _____ Юшин Е.В.

Руководитель к.т.н., доцент _____ Горячев А.В.

Консультант _____ Лебедева Т.Н.

Консультант по нормоконтролю _____ Родионова Е.А.

Санкт-Петербург
2023

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой САПР, к.т.н., доцент
_____ Бутусов Д.Н.
«___» _____2023 г.

Студент Юшин Е.В. Группа 9309

Тема работы: Разработка серверной части приложения для хранения паролей

Место выполнения ВКР: кафедра САПР, СПбГЭТУ «ЛЭТИ»

Исходные данные (технические требования):
Разработать внутренний программный интерфейс десктоп приложения, обеспечивающий связь с удалённым сервером хранения паролей, используя язык программирования C#, в среде разработки Visual Studio 22. Шифрование данных пользователя происходит с помощью алгоритма AES. Приложение должно включать в свой функционал: генерацию паролей, возможность поиска по записям, хранение в зашифрованном виде логинов и паролей пользователя, возможность редактирования этих ячеек, обеспечивать возможность хранения других персональных данных, иметь необходимый и достаточный уровень защиты от несанкционированного проникновения. Серверную часть необходимо реализовать с помощью фреймворка ASP.NET Core, связь БД и серверной части приложения должна осуществляться через Entity Framework Core. Приложение должно запускаться на windows 10 22H2.

Содержание ВКР: аннотация, введение, реферат, анализ существующих решений, анализ алгоритмов шифрования, проектная часть, экономическое обоснование (доп. раздел), заключение, список использованных источников (литературы), приложения.
Перечень отчетных материалов: пояснительная записка, иллюстративный материал, материалы к презентации ВКР, иные отчетные материалы.

Дополнительный раздел – Экономическое обоснование ВКР.

Дата выдачи задания «__» _____2023г.

Дата представления к защите «__» _____2023г.

Студент _____ Юшин Е.В.

Руководитель к.т.н., доцент _____ Горячев А.В.

Консультант _____ Лебедева Т.Н.

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой САПР, к.т.н., доцент

_____Бутусов Д.Н.

« » ____2023_ г.

Студент Юшин Е.В. Группа 9309

Тема работы: Разработка серверной части приложения для хранения паролей

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	11.04 – 19.04
2	Выполнение первой главы ВКР – «Анализ существующих программных средств»	20.04 – 27.04
3	Выполнение второй главы ВКР – «Проектная часть»	28.04 – 07.05
4	Выполнение дополнительного раздела «Экономическое обоснование»	08.05 – 10.05
5	Оформление пояснительной записки	11.05 – 17.05
6	Оформление иллюстративного материала	18.05 – 23.05
7	Прохождение предварительной защиты на кафедре	30.05
8	Представление ВКР к защите.	06.06

Студент _____ Юшин Е.В.

Руководитель к.т.н., доцент _____ Горячев А.В.

Консультант _____ Лебедева Т.Н.

РЕФЕРАТ

Пояснительная записка изложена на 56 стр. и содержит 19 рис., 5 табл., 11 ист., 2 прил.

РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ, АЛГОРИТМЫ ШИФРОВАНИЯ, АНАЛИЗ СОВРЕМЕННЫХ ПРОГРАММНЫХ СРЕДСТВ, LASTPASS, 1PASSWORD, BITWARDEN, MULTIPASSWORD, DASHLINE

В данной работе будет произведён анализ существующих программных средств. После проведенного анализа будут выявлены требования пользователей, образующий необходимый функционал.

Затем произведены теоретические исследования современных методов и алгоритмов криптографической защиты, на основе которого будет выбран алгоритм шифрования для последующей разработки, а также составлены рекомендации пользователям.

Затем начинается проектная часть разработки, во время которой будет предоставлена архитектура разрабатываемого ПО, структура базы данных и используемые во время разработки инструменты. Результатом данного раздела будет являться готовый программный продукт.

В последнем разделе данной работы будет представлено экономическое обоснование выполненной работы и разработанному программному продукту.

ABSTRACT

In this final qualifying work, the development of the server part of the password storage application was carried out, developed in accordance with modern user requirements identified based on the analysis of current software tools.

Also, in this work, theoretical research was carried out in the field of modern methods of information protection and cryptographic protection algorithms.

And calculations of the economic justification of the developed product were also carried out.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире существует огромное количество сервисов и приложений, для доступа к которым нам необходимо помнить свой логин и пароль. Как правило, люди используют либо один пароль ко всем учётным записям, либо несколько легко запоминаемых. Однако это всё не гарантирует безопасное пользование, ведь скомпрометировав пароль злоумышленник сразу получит доступ ко всем сервисам пользователя, либо, в случае легко запоминаемых паролей их могут просто подобрать, используя специальные программы.

Некоторые люди прибегают к использованию бумажного носителя, где записывают все данные от учётных записей, но безопасность и практичность этого метода в данном случае является спорной. Использование текстовых документов на одном из устройств, или хранение данных учётных записей в сообщениях мессенджера, также не является разумным с точки зрения безопасности, потому что, получив доступ к устройству, перед злоумышленником открываются двери во все учётные записи жертвы. Несмотря на существование двухфакторной аутентификации, требующей взаимодействия с ещё одним устройством или данные биометрии, вероятность несанкционированного доступа к аккаунту без связки логин-пароль + двухфакторная аутентификация является высокой.

Однако на помощь простым пользователям и работникам компаний приходят приложения – менеджеры паролей, обеспечивающие шифрование записанных пользователем данных от учётных записей, и повышающие безопасность и удобство пользователя.

В качестве объекта исследования в данной работе будут рассмотрены аспекты разработки и настройки серверной части приложения для хранения паролей, а также последующая эксплуатация.

В качестве предмета исследования обозначим разработку серверной части приложения для хранения паролей, то есть создание программного продукта, который позволит пользователям безопасно хранить и управлять своими паролями.

Данная работа является актуальной, потому что в современном цифровом мире, где даже у обычного пользователя присутствует огромное количество данных учётных записей, которые необходимо помнить и часто использовать. Как раз из-за этого большого количества логинов и паролей исходит множество проблем, которые решаются приложениями-менеджерами паролей. Поэтому проведённое исследование внесёт свой вклад в непомерно важное для всего цифрового общества дело.

Целью данной работы является разработку серверной части приложения для хранения паролей по выявленным в ходе анализа требованиям.

Для достижения цели необходимо выполнить следующие задачи:

- Проанализировать предметную область.
- Провести анализ существующих программных решений для хранения паролей.
- Изучить актуальные алгоритмы шифрования и методы защиты.
- Привести концепт - модель приложения.
- Разработать программное обеспечение.

1. АНАЛИЗ АКТУАЛЬНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

1.1 Классификация программного обеспечения для управления паролями

В современном мире безопасность является одним из наиболее актуальных и важных вопросов, которые требуют повышенного внимания. В связи с этим, все большую популярность получают программные средства для хранения паролей, так как они позволяют обезопасить персональные данные пользователей от несанкционированного доступа.

Данная глава будет основана на исследовании и анализе различных программных продуктов, предназначенных для хранения паролей, с целью выбора наиболее эффективного и надежного средства для реализации задачи.

Будут рассмотрены основные принципы функционирования программного обеспечения для управления паролями, классификация данных инструментов по различным критериям, а также проведён их сравнительный анализ с точки зрения функциональности, безопасности и удобства использования. Полученные результаты исследования помогут в понимании необходимых и достаточных функций, а также послужат основой для создания серверной части приложения для хранения паролей, соответствующего современным требованиям безопасности и удобства использования.

Менеджер паролей – приложение, облегчающее пользование другими приложениями или сервисами, работающими с учётными записями. Такое программное обеспечение (ПО) хранит логины и пароли пользователя в зашифрованном виде, используя для этого базу данных (БД), которая реализуется посредством удалённого сервера, либо файлы, остающиеся на компьютере пользователя. Как правило, такие приложения поддерживают автозаполнение и авто сохранение вводимых данных из учётных записей.

Зачастую их можно встретить в виде расширений или в стандартных функциях браузера.

Менеджеры паролей можно выделить в несколько категорий:

- Десктоп-приложения – обеспечивают обработку и хранение паролей пользователя, установленного на компьютере, и также поддерживают возможность ручного добавления данных учётных записей.
- Сетевые сервисы – обеспечивают хранение данных учётных записей, архивируя их, используя облачные сервисы, и обычно представленные web-версиями. Пример – Google Password, сохраняющий данные учётных записей пользователя в своём облачном хранилище.
- Портативные приложения – это приложения, устанавливаемые на переносные носители и обеспечивающие обработку и хранение данных учётных записей пользователя.

По типу хранения данных, эти программные средства делятся на следующие категории:

- Локальные приложения – приложения, обеспечивающие обработку и хранение сохранённых данных учётных записей, в зашифрованном виде на компьютере пользователя. Существенным минусом данного вида хранения информации является возможность потери всех учётных записей, в случае проникновения злоумышленником в компьютер.
- Облачные приложения – это приложения, шифрующие и отправляющие данные пользователя на выбранный удалённый сервер, такие приложения позволяют получать пользователям доступ с различных устройств, имеющих доступ в интернет. Но в

случае несанкционированного доступа к серверу – данные будут скомпрометированы.

Также, можно разделить приложения по открытости исходного кода:

- Приложения с открытым исходным кодом – то есть приложения, код которых доступен для просмотра любому пользователю, что является сравнительно безопасным вариантом, благодаря частым проверкам и сообществу пользователей, помогающим разработчикам быстро обнаруживать и устранять уязвимости.
- Приложения с закрытым исходным кодом – приложения, исходный код которых закрыт для просмотра. Такие приложения считаются менее безопасными, так как аудит безопасности производится только силами сотрудников компании, из-за чего они дольше могут оставаться незамеченными для разработчиков.

В целом, программные средства для хранения паролей — это важный инструмент для защиты данных пользователей. Однако, каждый пользователь должен выбирать приложение, которое наиболее соответствует его потребностям и уровню безопасности.

1.2 Анализ современных приложений для управления паролями

В данном параграфе будет приведён анализ современных десктопных программных средств по хранению и обработке данных учётных записей, и выявлен необходимый и достаточный функционал для последующей разработки серверной части приложения, удовлетворяющего запросы современного пользователя.

1.2.1 LastPass. Является одним из самых популярных десктопных приложений для хранения данных учётных записей, банковских карт, заметок, а также банковских аккаунтов. Данные шифруются симметричным aes-256

шифрованием и располагаются на серверах компании, из-за чего может возникать беспокойство за конфиденциальность хранимых данных. Для доступа к сохраненным паролям, то есть для входа в аккаунт, необходимо установить мастер-пароль. Но, как заверяют разработчики, мастер-пароль не отправляется на сервера компании и не хранится нигде в приложении. Также, благодаря использованию HMAC и RSA алгоритмов повышается сложность атаки на передаваемые данные.

Приложение LastPass проводит проверку устройства на вредоносное ПО, чтобы предотвратить несанкционированный доступ к админ-панели приложения.

LastPass обладает кроссплатформенностью, и работает на всех операционных системах, а также поддерживает работу с широким спектром устройств, включая персональные компьютеры, мобильные устройства и планшеты.

Основным преимуществом LastPass является автозаполнение форм логина и пароля на сайтах, настраиваемая по количеству символов генерация паролей.

Также можно выделить многофакторную аутентификацию и установку доверенных устройств, на которых, введение мастер пароля для входа не потребуется. В качестве доверенных устройств, можно установить даже флеш-накопители (рисунок 1) [1].

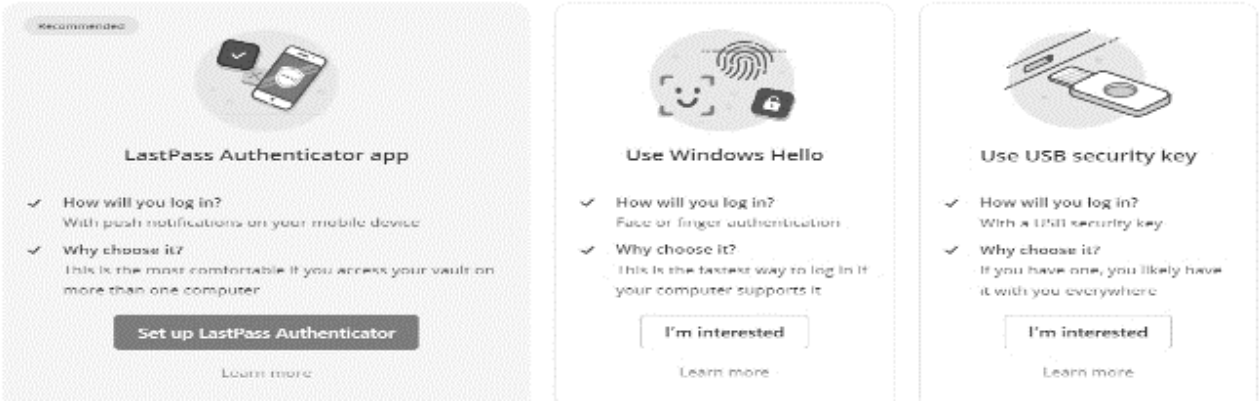

Рисунок 1 - Установка доверенных устройств в LastPass

Некоторые функции приложения доступны только в платной версии, например мультифакторная аутентификация. (рисунок 2).

Рисунок 2 - Разница в многофакторной аутентификации в зависимости от подписки

1.2.2 Приложение для хранения и управления паролями 1Password. Это десктопное приложение с закрытым исходным кодом для хранения логинов, паролей, реквизитов карт, адресов и номеров телефоном, а также других данных, для которых пользователь хочет конфиденциальности, доступные для добавления в хранилище данные представлены на рисунке 3. Данные шифруются с помощью 256 битного AES шифрования и зашифрованные данные локально хранятся на устройстве пользователя, доступ к которым скрыт за мастер-паролем аккаунта 1Password.

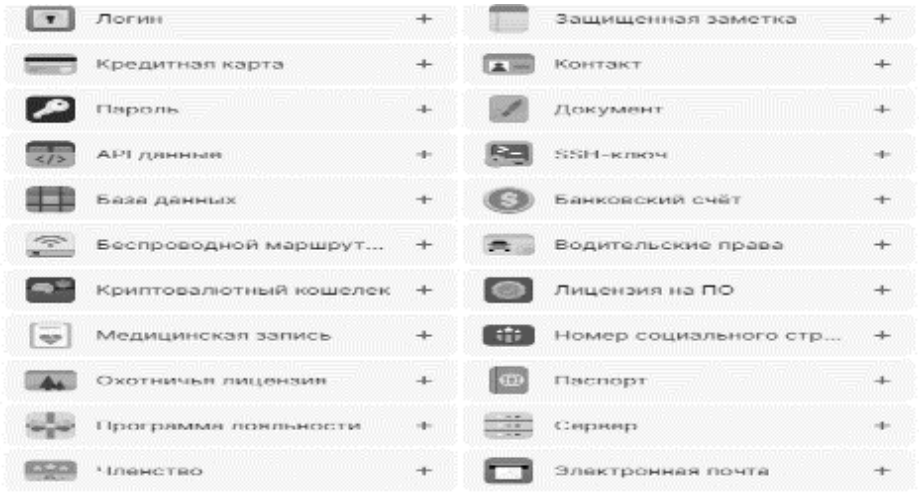

Рисунок 3 - Возможные для добавления в хранилище данные

Поддерживается кроссплатформенность, но отсутствует одновременный многопользовательский режим работы, что отчасти нивелируется возможностью поделиться сохранёнными данными с другими пользователями, при этом присутствует возможность группировки данных в один раздел и поделиться ссылкой на один этот раздел. Дополнительно возможна генерация паролей и имени пользователя и доступна функция автозаполнения форм данных учётных записей. Аналогично в 1Password имеется двухфакторная аутентификация с помощью мобильных устройств.

Благодаря встроенной функции Watchtower, пользователь может проанализировать безопасность, оценив надёжность своих паролей и проконтролировать наличие повторяющихся паролей (рисунок 4) [2].

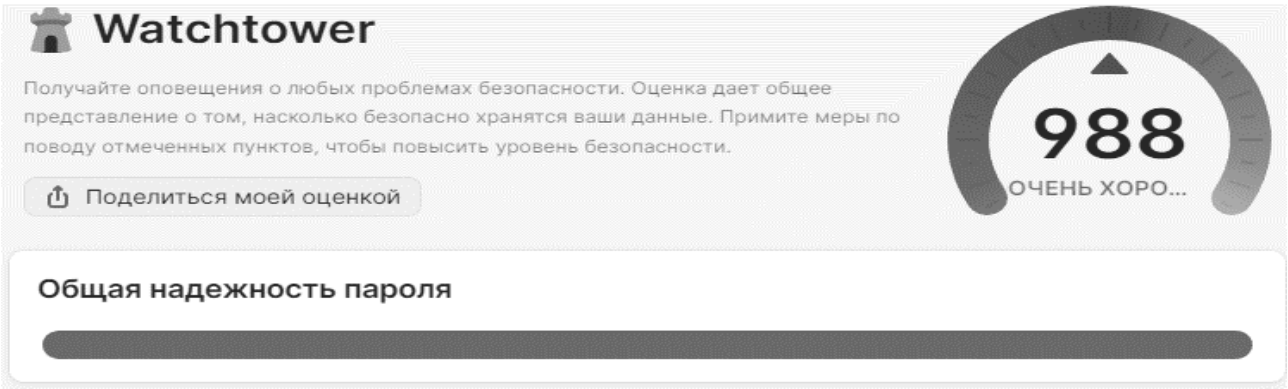

Рисунок 4 - Оценка надёжности пароля в 1Password

Приложение распространяется по программе подписки, с возможностью бесплатного пробного периода в 14 дней.

1.2.3 Dashlane. это приложение с закрытым исходным кодом для хранения личных данных, использующее шифрование aes-256, а затем отправляющее данные в зашифрованном виде на сервер компании Dashlane.

В приложении так же реализован генератор логинов и паролей, который позволяет пользователям создавать уникальные и безопасные пароли для каждого сайта.

Для дополнительной защиты Dashlane поддерживает включение двухфакторной аутентификации. Это означает, что пользователи будут должны ввести свой мастер-пароль и код, полученный на их мобильном устройстве, чтобы получить доступ к виртуальному хранилищу своих данных.

Dashlane также является кроссплатформенным приложением, что означает, что пользователи могут использовать его на разных устройствах и операционных системах. Пользователи могут синхронизировать свои данные между своими устройствами, чтобы иметь доступ к ним везде, где им это нужно.

Наконец, Dashlane поддерживает одновременный многопользовательский доступ, где каждый из пользователей может иметь собственную папку для хранения данных.

Для доступа ко всем этим функциям необходимо оформить подписку, но существует пробный 14-дневный период [3].

1.2.4 Bitwarden. Это приложение, помогающее хранить и использовать логины, пароли, реквизиты карт, и другую личную информацию, использующее сквозное 256 AES-CBC шифрование и хранящее хешированные с солью данные на удалённом сервере компании. Таким образом для получения доступа к

хранилищу данных необходимо ввести в качестве логина адрес электронной почты и мастер-пароль**.**

В виде идентификатора каждому пользователю присваивается уникальная фраза отпечатка, то есть последовательность случайных слов на английском. Это помогает во время авторизации на новых устройствах или с восстановлением аккаунта.

Приложение поддерживает кроссплатформенность и синхронизированную связь между различными устройствами.

Из стандартных функций, в приложении имеется:

- автозаполнение форм,
- двухфакторная аутентификация, в том числе с использованием биометрии,
- генерация логина и пароля,
- синхронизация хранилища, которая помогает переносить пароли с одного устройства на остальные.

Для переноса сохранённых данных дополнительно присутствует возможность экспорта, с помощью передачи зашифрованного файла.

Bitwarden является одним из примеров приложений с открытым исходным кодом, благодаря чему аудит безопасности может провести любой пользователь, и убедится в прозрачности и безопасности программы.

Bitwarden предлагает постоянную бесплатную версию со всеми функциями и с неограниченным количеством устройств. Но существуют варианты премиум подписки, расширяющих стандартные функции приложения и добавляющие несколько новых, таких как продвинутая аутентификация, аварийный доступ и отчёты о безопасности [4].

1.2.5 MultiPassword. Это кроссплатформенное программное обеспечение с закрытым исходным кодом для управления паролями, позволяющее хранить и обрабатывать данные для различных учётных записей.

Являясь относительным новичком на рынке ПО, multipassword использует те же алгоритмы шифрования, что и конкуренты, для шифрования данных на стороне пользователя – сквозное AES-256, а для передачи зашифрованных данных пользователя серверу с помощью PBKDF2, RSA и HKDF. Надёжность от взлома обеспечивается обязательной двухступенчатой аутентификацией, то есть необходимостью кроме ввода мастер-пароля, также вводить секретный цифровой ключ. Также, отсутствует возможность восстановления мастер-пароля, но его изменение возможно при успешной авторизации в аккаунт.

Здесь пользователю представляется возможность создания сложных настраиваемо-генерируемых паролей (рисунок 5), автозаполнение форм, как вводимых пользователем, то есть занесение данных сразу в базу данных приложения, так и автозаполнение веб-форм ранее сохранёнными данными.

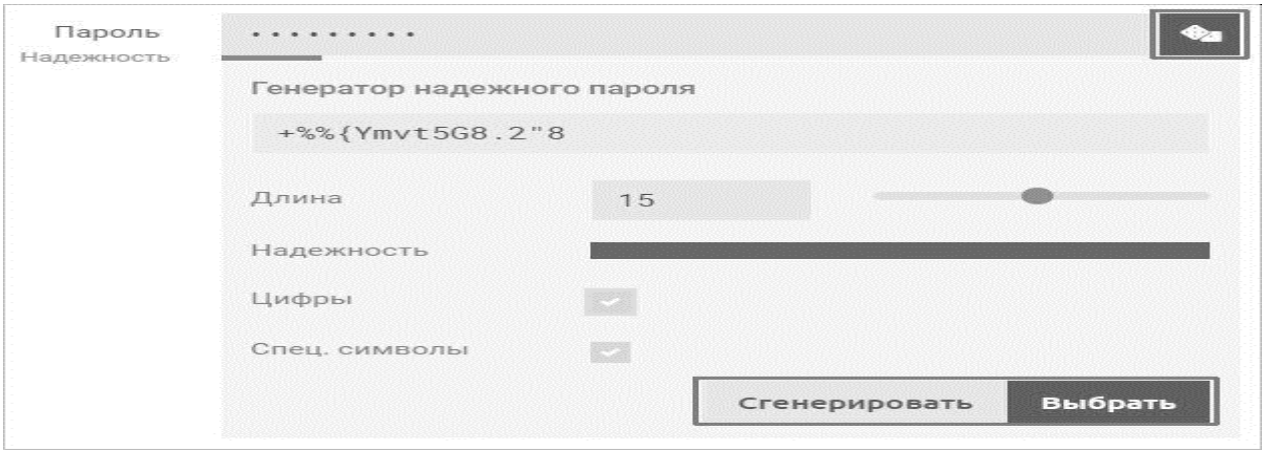

Рисунок 5 - Параметры генерации пароля

Поддерживается импорт паролей из других менеджеров паролей и браузеров, а также экспорт уже сохранённой информации в виде табличного файла.

Встроенная функция анализа надёжности хранимых паролей поможет пользователю повысить свою безопасность и защититься от атак перебором на пароли.

Приложение распространяется по подписке, с возможностью бесплатного пробного периода в 30 дней [5].

Таблица1 - Сравнение характеристик рассмотренных приложений

	LastPass	1Password	Dashlane	Bitwarden	MultiPassword
Открытый код	нет	нет	нет	да	нет
Платформы	Windows, MacOS, Linux, Android, IOS, а также браузеры	Windows, MacOS, Linux, Android, IOS	Windows, MacOS, Linux, Android, IOS	Windows, macOS, Linux, iOS, Android, и браузеры Chrome, Firefox, Safari, Edge и Opera.	Windows, macOS, Linux, iOS, Android, и браузеры Chrome, Firefox
Хранение данных в облаке	да	да	да	да	да
Двухфакторная аутентификация	да	да	да	да	да
Одновременный многопользовательский доступ	да	нет	да	да	нет
Форма распространения лицензии	Бесплатно + подписка	бесплатно 14 дней, подписка	бесплатно 14 дней, подписка	Бесплатно + подписка	Подписка, бесплатно 30 дней
Отличие премиум-версии от обычной	Многопользовательский доступ, расширенные возможности аутентификации, поддержка использования больше 1 устройства и их синхронизация	Присутствует только премиум версия	Присутствует только премиум версия	Продвинутая аутентификация, аварийный доступ и отчёт безопасности	Присутствует только премиум версия

После приведённого выше сравнительного анализа, обобщённого на таблице 1, можно заключить, что современное программное обеспечение для обработки данных учётных записей, как правило:

- Обладает не просто кроссплатформенностью, а имеет версию непосредственно для каждой операционной системы, включая

мобильные ОС, а некоторые приложения так же представлены web-версиями или расширениями браузера;

- Поддерживает использование на различных устройствах и синхронизацию между ними;
- Поддерживает двухфакторную аутентификацию;
- Имеет встроенные возможности для импорта/экспорта данных;
- Обладает встроенным настраиваемым генератором паролей и имён пользователя;
- Поддерживает функцию автозаполнения и автосохранения форм у учётных записей пользователя;
- Поддерживает одновременный многопользовательский доступ;
- Обладает хорошей криптографической защитой, соответствующей современным стандартам (в особенности при наличии открытого исходного кода)
- Хранит зашифрованные данные в облаке компании разработчика, либо предоставляет пользователю возможность выбрать сервер хранилища самостоятельно.

В заключении, можно сделать вывод, что современное конкурентноспособное программное обеспечение по управлению паролями должно предоставлять пользователям перечисленные выше функции по разумной цене, и иметь постоянные обновления безопасности, с чем поможет сообщество пользователей, при условии, что открыт исходный код приложения.

Таким образом можно определить требования, выдвигаемые к разработке ПО:

- Поддержка хотя бы одной из ведущих операционных систем.
- Взаимодействие с удалённым хранилищем.

2 АНАЛИЗ СОВРЕМЕННЫХ МЕТОДОВ И АЛГОРИТМОВ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ

2.1 Теоретическое исследование современных методов криптографической защиты

Криптографическая защита имени пользователя и пароля в приложения является одной из наиболее важных задач в области информационной безопасности. Существует множество методов и алгоритмов, которые используются для защиты информации, однако не все методы обеспечивают высокий уровень безопасности. Криптографическая защита паролей является одним из ключевых механизмов, который позволяет защитить данные пользователей от несанкционированного доступа и кражи.

В современных методах защиты можно выделить:

2.1.1 Хеширование паролей. Является одним из наиболее распространённых методов криптографической защиты информации. Хеш-функция принимает на вход произвольное количество данных и возвращает фиксированную длину хеш-значения. Хеш-значение является уникальным аутентификатором для каждого набора данных, что позволяет использовать это значение для проверки целостности данных.

Суть метода заключается в том, что при регистрации пользователя его пароль хешируется и сохраняется в базе данных. Во время аутентификации в аккаунт система сравнивает хеш-значение сохранённого пароля с хеш-значением введённого пользователем пароля. При совпадении этих значений, пользователь получает доступ в аккаунт.

Однако хеширование данных не является идеальным методом криптографической защиты. Проводящий атаку злоумышленник может использовать таблицы с заранее вычисленными хеш-значениями (так

называемые «rainbow tables»), для быстрого поиска соответствующего пароля. Для уменьшения вероятности успешной атаки с помощью таких таблиц, разработчики применяют хеширование данных с помощью «соли» - случайной строки, добавляющейся к паролю перед хешированием.

2.1.2 Хранение паролей в виде шифрованных текстов. Ещё одним часто встречающимся методом защиты информации с помощью криптографии, является шифрование паролей с помощью различных алгоритмов шифрования и последующее сохранение в базе данных. Во время аутентификации пользователя система проводит расшифровку сохранённого пароля и сравнивает его с паролем, введённым пользователем.

2.1.3 Использование многофакторной аутентификации. Методом многофакторной аутентификации – это метод, при котором для доступа к системе требуется преодолеть несколько рубежей защиты. Например, в дополнении к стандартной аутентификации требуется также ввести код подтверждения с другого закреплённого устройства или же использовать биометрию.

Этот метод обеспечивает относительно высокий уровень безопасности, потому что атакующему злоумышленнику необходимо будет скомпрометировать несколько факторов аутентификации. Однако, подобный вид аутентификации может быть крайне неудобным для пользователя, из-за чего данный метод защиты может иметь низкую частоту использования.

2.1.4 Использование сложных и длинных паролей. По статистике, большое количество пользователей используют простые и легко запоминающиеся последовательности символов в качестве паролей. Но такие пароли обладают низким уровнем надёжности, потому что злоумышленник может воспользоваться методом подбора паролей по словарям, состоящим из ранее скомпрометированных паролей других пользователей. Таким образом, с точки зрения, как пользователей, так и разработчиков, необходимо

использовать сочетание цифр и символов, включая специальные символы. Длина таких сочетаний должна быть не менее 10.

Таким образом комбинирование многофакторной аутентификации вместе с несколькими другими методами, должно обеспечивать высокий уровень надёжности. Но обычным пользователям не стоит забывать, о необходимости ответственного подхода к безопасности собственных данных. Речь идёт не только про использование сложных паролей, но и про аккуратное потребление информации в сети.

2.2 Теоретическое исследование алгоритмов шифрования данных в современных приложениях

Современные приложения для управления паролями должны обеспечивать высокий уровень безопасности, чтобы защитить конфиденциальные данные пользователей от несанкционированного доступа. Одним из ключевых механизмов обеспечения безопасности в таких приложениях является шифрование данных. Шифрование позволяет защитить пароли и другие конфиденциальные данные от перехвата и расшифровки злоумышленниками.

Существует множество алгоритмов шифрования, которые могут использоваться в приложениях для управления паролями. Каждый из них имеет свои преимущества и недостатки, и выбор конкретного алгоритма зависит от требований приложения и уровня безопасности, который необходимо обеспечить.

Современные алгоритмы шифрования данных классифицируются на следующие четыре категории:

- Симметричные алгоритмы;

Симметричные алгоритмы шифрования – это алгоритмы, использующие один и тот же ключ для шифрования и дешифрования информации. Эти алгоритмы основаны на математических преобразованиях, которые изменяют данные таким образом, что дешифрование и использование становится невозможным без знания ключа.

Примером популярного симметричного алгоритма является:

2.2.1 AES. Advanced Encryption Standard является одним из самых распространённых алгоритмов симметричного шифрования. Он представляет собой блочное шифрование с размеров блока в 128 бит и ключом размером 256 бит.

Алгоритм AES-256 состоит из нескольких итераций, на каждой из которой применятся несколько операций. Операции включают в себя разбиение блока как матрицы байтов и применение к нему линейных и нелинейных преобразований.

Преимуществами AES-256 являются высокий уровень безопасности, обеспечивающийся благодаря большому размеру ключа и сложному алгоритму шифрования. AES также же является быстрым и эффективным алгоритмом.

Недостатками данного алгоритма являются ограничения на использование для больших объёмов данных и высокие требования к оборудованию для расшифровки, что может повышать затраты на хранения и обработку данных. Также, существует вероятность атаки методом перебора ключей, но на практике это требует большого количества времени и вычислительной мощности.

Для использования симметричного алгоритма шифрования, отправитель и получатель должны заранее договориться о ключе. Ключ также необходимо передать в зашифрованном виде через другой канал связи.

Использовав ключ для шифрования данных, отправитель пересылает их получателю, который применив этот же ключ проводит расшифровку [6].

- Ассиметричные алгоритмы;

Ассиметричные алгоритмы шифрования, также известные как криптография с открытым ключом, используют два разных ключа для шифрования и дешифрования информации: открытый и закрытый.

Открытый ключ используется для шифрования отправляемых данных, но только владелец закрытого ключа может его расшифровать.

Пример ассиметричного алгоритма шифрования:

2.2.2 RSA. Представляет собой асимметричный криптографический алгоритм, который используется для защиты передачи данных по сети. Он был предложен в 1977 году Ривестом, Шамиром и Адлеманом и остается одним из самых популярных методов шифрования**.**

Основная идея алгоритма заключается в использовании математических принципов для генерации ключей шифрования и расшифровки данных.

Из всех преимуществ алгоритма RSA можно выделить отсутствие необходимости обмениваться секретными ключами шифрования, как при использовании симметричных алгоритмов. Дополнительно алгоритм RSA допускает возможность использования цифровых подписей, то есть поддерживает создание и использование цифровых подписей у документов, что помогает в проверке подлинности данных. Также RSA обеспечивает высокий уровень безопасности передаваемых данных, так как обладает устойчивостью к атакам методом подбора или взломам.

Главными недостатками алгоритма RSA являются медленная скорость работы и высокие требования к вычислительным ресурсам. Такие требования появляются из-за подбора и использования больших простых чисел, необходимых при генерации ключей. Однако даже этот алгоритм подвержен атакам типа Man-in-the-middle, во время которых злоумышленник встраивается между отправляющими сторонами из-за чего может перехватывать и изменять сообщения между отправителем и получателем [6].

- Хеш-функции

Хеш-функция – представляет собой математическую функцию, которая преобразовывает любую входную строку произвольной длины в строку фиксированной длины.

Шифрование с помощью хеш-функций работает следующим образом: сначала исходное сообщение преобразуется с помощью хеш-функции в строку с фиксированной длиной, называемую хешем. Затем полученную строку используют для передачи данных, а исходное сообщение удаляется или заменяется другим.

Главными преимуществами использования хеш-функций можно назвать целостность данных, так как при изменении исходного текста, изменится и его хеш-функция, что облегчает выявление подмены данных. Также шифрование с помощью хеш-функций обладает высокой скоростью и эффективностью при работе с большими объёмами данных.

Однако, использование хеш-функций не обладает высокой надёжностью против атак методом подбора, во время которых атакующий перебирает все возможные комбинации, в попытке найти подходящий хеш.

Поэтому, для повышения безопасности применяют алгоритмы с секретным ключом (например PBKDF2). Эти алгоритмы позволяют создавать и использовать хеши, зависящие от секретного ключа, неизвестного для атакующего.

2.2.3 PBKDF2. Password-Based Key Derivation Function 2 – это алгоритм, используемый для производства ключей из паролей. Является итеративным алгоритмом, применяющим хеш-функцию к паролю с добавлением «соли», также являющейся итерированной. Результатом объединения этих операций является конечный ключ.

Основой работы этого PBKDF2 является функция хеширования, которую допускается выбрать самостоятельно, но чаще всего используются SHA-1 [6].

Главными преимуществами алгоритма являются его простота в использовании и высокий уровень безопасности, обеспечивающийся благодаря итеративному процессу хеширования, который применяется к паролю и соли.

Недостатками PBKDF2 можно назвать очень ресурсоёмкий процесс итерации, требующий значительные вычислительные мощности.

- Гибридные алгоритмы шифрования

Гибридные алгоритмы шифрования представляют собой комбинацию нескольких различных алгоритмов шифрования. Идея подобного использования заключается в том, чтобы использовать преимущества разных схем шифрования и создать более надёжное решение, чем каждый из компонентов по отдельности. Гибридные алгоритмы шифрования делятся на следующие категории:

- Сложная криптографическая защита

Это защита, которая сводится к использованию нескольких алгоритмов на каждом этапе процесса шифрования.

- Множественное хеширование

Представляет собой использование комбинаций хеш-функций, позволяющих гарантировать целостность и контрольную сумму данных.

- Низкоуровневое шифрование

Представляет собой использование аппаратных средств для процесса шифрования.

Преимуществами гибридных алгоритмов шифрования являются очень высокий уровень безопасности и устойчивость к атакам, гибкость, позволяющую найти решение, удовлетворяющее конкретным требованиям безопасности.

Однако несмотря на все преимущества, гибридные алгоритмы шифрования обладают целым рядом недостатков:

- Вычислительная сложность

Гибридные алгоритмы используют несколько алгоритмов, что требует большего количества вычислительных ресурсов, чем простые алгоритмы.

- Сложность управляемости

Гибридные алгоритмы шифрования являются менее управляемыми, чем простые методы, потому что требуют различных ключей для различных алгоритмов.

- Уязвимость к смешиванию

В случае если один из алгоритмов будет скомпрометирован, это может создавать уязвимости во всей системе шифрования.

В современных приложениях используются разнообразные алгоритмы шифрования данных, такие как симметричные, асимметричные, использования хеш-функций и комбинацию всех алгоритмов.

Алгоритмами, обеспечивающими наибольший уровень надёжности, являются гибридные алгоритмы шифрования. Однако каждый алгоритм шифрования имеет свои недостатки и может быть взломан. Поэтому важно использовать сильные алгоритмы шифрования данных и регулярно их обновлять.

Также приложения должны обеспечивать защиту, используя современные методы криптографической защиты, такие как многофакторной аутентификации и хранение паролей в зашифрованном виде.

В результате анализа предметной области исследования были даны основные определения и классификация программных средств для хранения паролей, которая включала в себя классификацию по типу поддерживаемых платформ, способу хранения данных и открытость исходного кода приложения. Из анализа существующих средств сделан вывод о необходимости разработки десктоп приложения с хранением данных на удалённом сервере и открытым исходным кодом.

Таким образом примерная первичная архитектура разрабатываемого программного обеспечения представляет собой API, связанный с удалённым сервером или облаком, обеспечивающим хранение зашифрованных пользовательских паролей. Предварительная модель логики представлена на рисунке ниже.

Рисунок 6 – Модель архитектуры всего приложения

При изучении современных методов криптографической защиты, такие как хранение паролей в зашифрованном виде, хеширование были рассмотрены существующие стандарты, выявлены их преимущества и недостатки. Также были рассмотрены актуальные алгоритмы шифрования такие как AES-256, SHA-256, RSA. Каждый из рассмотренных алгоритмов имеет свои сильные и слабые стороны, однако существует возможность комбинировать несколько алгоритмов шифрования вместе, что усиливает защиту, но добавляет трудностей в реализацию и управление такими системами.

Было решено использовать в дальнейшей разработке алгоритм AES, а в перспективе дальнейшего развития продукта существует возможность комбинирования данного алгоритма и каким-нибудь другим.

3. ПРОЕКТНАЯ ЧАСТЬ

3.1. Краткое описание средств реализации

3.1.1 ASP.NET Core. Представляет собой кроссплатформенный веб-фреймворк, предоставляющий разработчикам инструменты и библиотеки для создания веб-приложений с высокой производительностью. Своим функционалом данный фреймворк позволяет интегрировать различные пользовательские интерфейсы и сервисы. Также присутствует возможность подключения промежуточного программного обеспечения для участия в обработке запросов с клиентской части приложения. Дополнительно присутствует возможность выбора различных веб-сервисов для функционирования приложения. Присутствует поддержка установки дополнительных пакетов при помощи менеджера пакетов NuGet. Является кроссплатформенным решением.

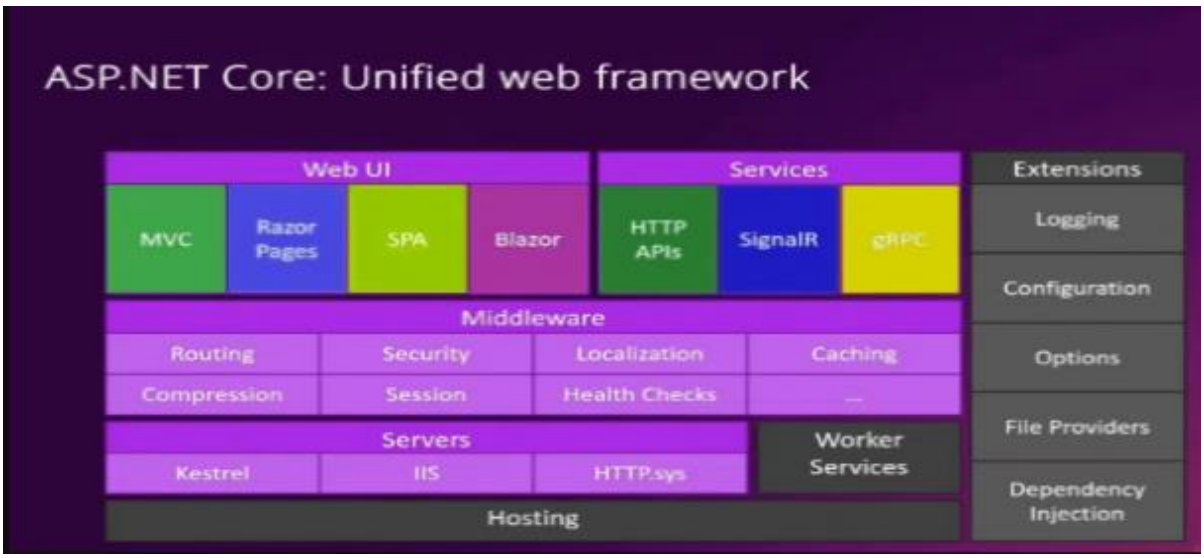

Рисунок 7 – Структура ASP.NET

3.1.2. ***Entity Framework***. Представляет собой технологию, позволяющую разработчикам работать с базами данных в виде объектов, представленных моделями. Entity Framework позвляет создавать модели данных, основанных на

уже существующих базах данных или создавать новые на основе моделей. С помощью Entity Framework можно добавлять, удалять и обновлять данные в базе данных. В целом, этот фреймворк упрощает работу с базами данных, а также обладает повышенной производительностью за счёт автоматической генерации sql-запросов. Поддерживает кроссплатформенный запуск на различных устройствах, добавляется в проект через установку из менеджера пакетов NuGet.

В целом, Entity Framework Core представляет собой связующее звено между базой данных и внутренним программным интерфейсом приложения, схема их взаимодействия представлена на рисунке 8 ниже.

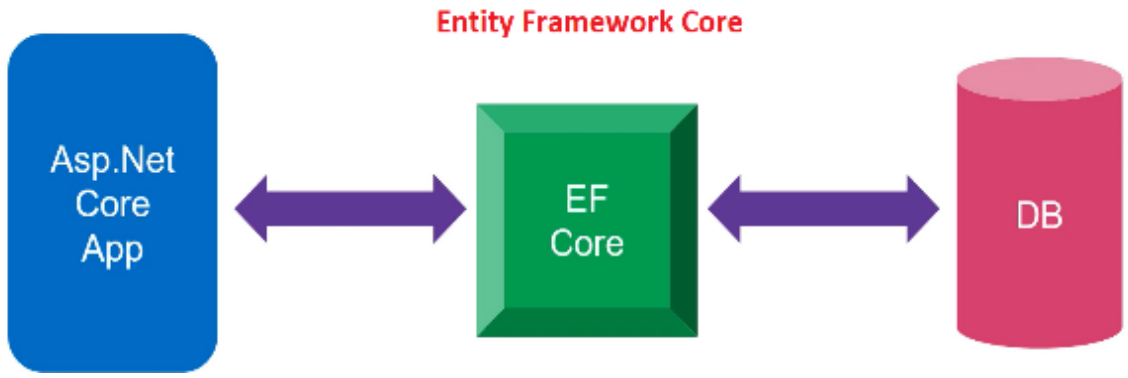

Рисунок 8 – Схема взаимодействия EF Core

В Entity Framework Core происходит представление объектов базы данных как моделей, то есть первоначально создаётся объект класса с полями, служащими свойствами, по которым в последствии будет происходить заполнение соответствующих ячеек базы данных. Пример такого класса UserData с полями Id, ServiceName, CardNumber, CardCVV, Description, Login и Password.

Листинг 1 – Класс модели данных

```
public partial class UserData
{
    public int Id { get; set; }
    public string ServiceName { get; set; }
```

```
        public string Login { get; set; }
        public string Password { get; set; }
        public string? CardNumber { get; set; }
    public string? CardCVV { get; set; }
    public string? Description { get; set; }
    }
```

Затем требуется создать контекст для базы данных, то есть определить множество объектов этой базы данных, создать файл и определить его имя.

Листинг 2 – Определение контекста базы данных

```
namespace DBRequest
{
    // Создаём контекст для базы данных
    public partial class UserDataContext : DbContext
    {
        // Определяем множество объектов этой базы данных
        public DbSet<UserData> UserDates { get; set; } =
null!;
        // Создаём файл если его не существует
        public UserDataContext() => Database.EnsureCreated();
        // Определяем имя файла
        protected override void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite("Datasource =
DataPass.db;");
        }
    }
```

3.1.3 MS sqlite. Это легковесная реляционная система управления базой данных, которая используется в приложениях .NET. Включает в себе простой и удобный способ хранения и управления ячейками данных, обеспечивая при этом высокую производительность и низкие требования. SQLite использует язык SQL (язык запросов), с некоторыми дополнениями и опущениями. Так

SQLite лишена возможности многопользовательского управления, но поддерживает хранением базы данных в одной файле и обладает высокой скоростью записи. Эта база данных может быть использована для различных задач, таких как хранение пользовательских данных, кеширование результатов запросов, хранение данных конфигураций и т.д. Также возможно использование для хранения данных из приложения без доступа к сети. В целом Sqlite представляет собой встраиваемую систему управления базой данных, то есть является библиотекой, объединяющуюся с программой в результате чего становясь её частью. Перед началом работы с данным фреймворком, необходимо добавить его в проект, установив, используя менеджер пакетов NuGet.

3.2. Архитектура разрабатываемого программного обеспечения

Схема работы цельного приложения для хранения паролей расположена ниже на рисунке 9.

Рисунок 9 – Схема работы приложения

Таким образом на стороне пользователя в клиентской части приложения будет генерироваться запрос, обрабатываемый разрабатываемым в данной работе внутренним программным интерфейсом и затем отправляться на удалённый сервер за ответом, при получении ответа, проделав обратный путь ответ дойдёт до пользователя.

Пример сохранения логина и пароля при использовании пользователем цельного приложения:

- Пользователь вводит данные в поля логина и пароля в клиентской части приложения;
- Введённые данные считываются оттуда во временные переменные;
- Временные переменные зашифровываются алгоритмом AES-256;
- Зашифрованные переменные заполняют поля базы данных;
- Заполненный файл базы данных отправляется на сервер, где дополняются недостающие поля базы данных.

3.3.Структура базы данных и описание её объектов

В архитектуре приложения присутствуют несколько объектов базы данных. Первый объект, взаимодействующий с пользователем – Registrations, имеющий поля:

- Id типа Integer, обозначает порядковый номер записей и служит для облегчения структуризации ячеек;
- Regmail типа String, обозначает введённый пользователем логин для последующей авторизации;
- RegPass типа String, обозначает введённый пользователем пароль для последующей авторизации.

Следующий объект, взаимодействующий с пользователем и связанный с Registrations это – Autorizations, он имеет следующие поля:

- Id типа Integer, обозначает порядковый номер записей и служит для облегчения структуризации ячеек;
- Logmail типа String, обозначает введённый пользователем логин для проверки авторизации в окне авторизации;

- MasterPassword типа String, обозначает введённый пользователем пароль для проверки авторизации в окне авторизации.

Связь всех объектов базы данных показана на рисунке 10 ниже.

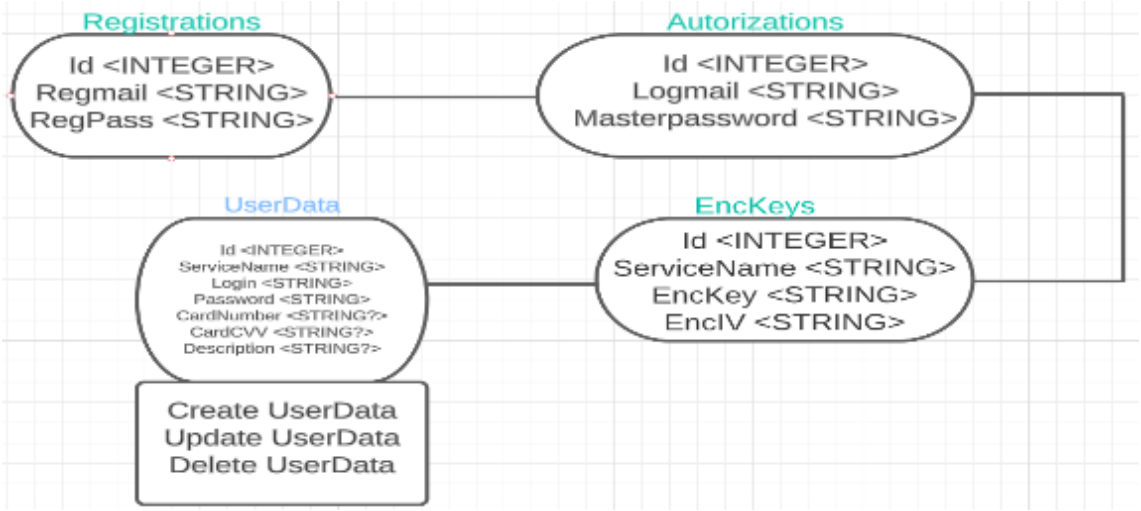

Рисунок 10 – Связь объектов БД

Объект базы данных EncKeys имеет следующие поля:

- Id, типа Integer, хранящий номер записи и служащий для облегчения стуктуризации;
- ServiceName, типа String, хранящий имя пользовательского сервиса, данные от которого в зашифрованном виде передаются в объект UserData;
- EncKey, типа String, хранящий сгенерированный ключ для AES-256 шифрования;
- EncIV. типа String, хранящий сгенерированный вектор инициализаци для AES-256 шифрования;

Объект базы данных UserData имеет следующие поля:

- Id, типа Integer, хранит порядковый номер записи;

- ServiceName, типа String, хранит имя пользовательского сервиса, необходим для соответствия пары login/password или cardnumber/cardcvv;
- Login, типа String, хранит зашифрованный логин, введённый пользователем;
- Password, типа String, хранит зашифрованный пароль, введённый пользователем;
- CardNumber, типа String?, хранит зашифрованный номер карты пользователя, по умолчанию имеет значение NULL, то есть является необязательным полем;
- CardCVV, типа String?, хранит зашифрованный cvv карты пользователя, по умолчанию имеет значение NULL, то есть является необязательным полем;
- Description, типа String?, хранит текстовую информацию для возможности добавления пояснения предыдущих пунктом пользователем.

Таким образом, введённые пользователем данные во время регистрации сохранятся в отдельный файл и во время авторизации произойдёт сравнение сохранённых данных с данными, введёнными в поля окна авторизации.

Кроме этого, при попытке пользователя сохранить логин и пароль от какого-то личного аккаунта, сгенерированные ключ шифрования и вектор инициализации будут сохранены в отдельный файл, а сами логин и пароль в зашифрованном виде будут сохранены в UserData, который затем будет загружен на сервер.

При поступлении запроса со стороны пользователя о предоставлении ему сохранённого пароля от определённого сервиса, в случае наличия такой ячейки, ему будут отправлены зашифрованные данные из файла UserData, к ним будет

найдена пара ключ-вектор из файла, хранящего ключ и вектор, а затем показано на экране клиентской части приложения в соответствующих полях.

3.4 Описание модулей и их реализация

В данном разделе будут приведено описание различных модулей программ и их реализация, а также продемонстрирована их работа по отдельности.

3.4.1 Класс Genpassword. Отвечающий за реализацию возможности настраиваемой генерации последовательности символом, подходящую, как для логина, так и для пароля. Состоит из он трёх методов:

- Public int GenParameters

Представляет собой метод, получающий на вход переменную типа Integer, обозначающую длину генерируемой последовательности. Необходим для определения с типом символов, используемых в генерации, то есть позволяет выбрать либо прописные буквы, либо заглавные, а также цифры и символы. Результатом работы возвращает переменную типа Integer, с кодом с информацией об выбранном типе символов.

- Public string TypeSymbPass

Представляет собой метод, получающий на вход переменную типа Integer из прошлого метода, свидетельствующую о типе выбранных символов для генерации последовательности. Результатом работы метода будет выведена последовательность символов, выбранная для генерации.

- Public string GenPass

Представляет собой метод, принимающий на вход переменную типа Integer, необходимую для определения длины генерируемой последовательности, а также последовательность символов из метода TypeSymbPass, результатом работы возвращает переменную типа String,

представляющую собой сгенерированную последовательность символов, заданной длины. Пример вызова методов из класса Genpassword представлен на рисунке 11 ниже

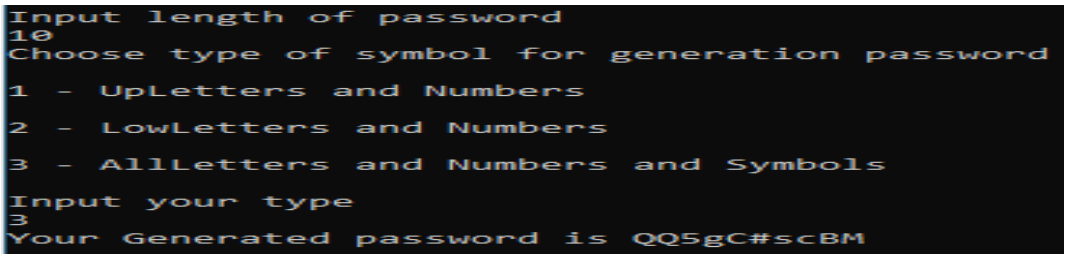

Рисунок 11 – Сгенерированная последовательность символов

3.4.2 Класс AESEncryptionManager. Реализует возможность шифрования и дешифрования любых переменных типа string. Состоит из двух методов:

- Public string Encrypt

Представляет собой метод, принимающий на вход переменную типа string, представляющую строку, которую необходимо зашифровать, а также переменные key и iv, типа string, обозначающие сгенерированные классом Genpassword ключ и вектор инициализации соответственно. Данный метод реализует шифрование строки, ключом и вектором инициализации, в результате работы возвращает переменную типа byte [], которая конвертируется в string.

- Public string Decrypt

Представляет собой метод, принимающий на вход три переменные типа string, переменную ciphertext, представляющую собой последовательность символов, полученную в результате работы прошлого метода, а также переменные key и iv, являющиеся ключом и вектором инициализации соответственно. Результатом работы возвращает переменную типа string, то есть расшифрованную последовательность из прошлого метода.

Пример вывода результата работы AESEncryptionManager и ключа с вектором инициализации представлен на рисунке 12 ниже.

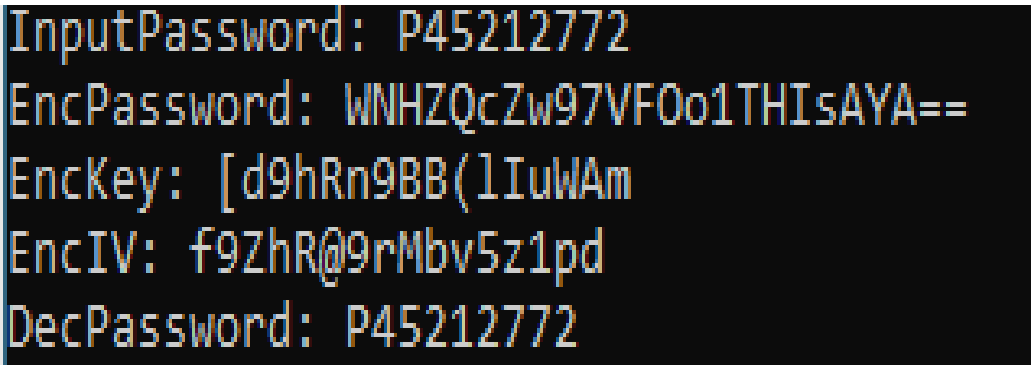

Рисунок 12 – Пример вывода результата вызова методов класса AESEncryption

Использование Entity Framework Core подразумевает объявление классов моделей базы данных, а также создание контекста базы данных, для последующего облегчённого взаимодействиями с ними.

3.4.3 Класс DatabaseContext. Представляет собой определение контекста базы данных, для последующего облегчённого взаимодействия и автоматической генерации sql-запросов. Содержит внутри себя:

- Определение множества используемых объектов.

То есть создаёт множество UserDates объектов UserData, для облегчённого доступа к полям последнего.

- Создание и определение файла базы данных, а также настройку сервера БД.

Пример сгенерированного файла БД с добавленными пользователем данными, приведён на рисунке 13 ниже:

	Id	ServiceName	Login	Password	CardNumber	CardCVV	Description
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Steam	/ZsvnmJksqiquIqQqBAlGQ==	YxeNRkHtUkh4TVCI/C8KyA==	NULL	NULL	NULL
2	2	Vk	log1	pass1	NULL	NULL	NULL
3	3	OK	siZ8bjijeBRCsZAUwdVtMw==	AUVNAcnwCYmHwypAR7Hrrw==	NULL	NULL	NULL
4	4	Test4	log2	pass2	NULL	NULL	NULL
5	5	Test5	lyAJKeYrL6LvZfdm2tQCwQ==	i4yXjxqSxikof91EZL3tPQ==	NULL	NULL	NULL
6	6	Test6	log3	pass3	NULL	NULL	NULL
7	7	Test7	ApzCdF1yePdDRcf3y8aePA==	WNHZQcZw97VFOo1THIsAYA==	NULL	NULL	NULL
8	8	Test8	log4	pass4	NULL	NULL	NULL

Рисунок 13 – Пример файла БД с сохранёнными пользовательскими данными

3.4.4 Класс RegContext. Представляет собой тоже самое определение контекста базы данных, но множеством объектов являются данные пользователя, указанные при регистрации. Необходим для сохранения логина и мастер пароля и последующей авторизации. Пример файла БД, содержащие логин и пароль пользователя, указанные при регистрации в приложении, представлен на рисунке 14 ниже:

	Id	Regmail	RegPass
	Фи...	Фильтр	Фильтр
1	1	uv9rsIz1li4f5Q1GnO8DeA==	3OSrI5SmQ1aDIxI102UExg==

Рисунок 14 – Пример файла БД с регистрационными данными пользователя

3.4.5 Класс RegKey. Представляет собой очередное определение контекста базы данных, однако множество объектов формируется из моделей, поля которых являются переменными, хранящими ключ шифрования и вектор инициализации, а также имя сервиса. Ключ и вектор необходимы для шифрования и дешифрования, а имя сервиса является ключом, по которому связываются пары зашифрованных строк и ключей. Пример файла БД, заполненного автоматически приведён на рисунке 15 ниже:

	Id	ServiceName	EncKey	EncIV
	Фи...	Фильтр	Фильтр	Фильтр
1	1	Steam	sqgh#!@VgdsC4]Kj	b@DR=-#1mP3GoiA
2	2	Registration	sqtghVgosgAC4]ZI	x@DRLhq]mP3GlH$R
3	3	Test3	&i)CzZ4`TX$glTn&	Q~+nfe+7Z`8mX}p7
4	4	Test4	VqMWa)!M1y5Ir3wI	_Nm_-}5BB%r~vlZf
5	5	Test5	[d9hRn9BB(lIuWAm	f9ZhR@9rMbv5z1pd

Рисунок 15 – Файл БД, хранящий ключ и вектор

3.4.6 Класс AutorizationController. Представляет собой контроллер данных, необходимый для считывания введённых данных пользователем, для прохождения авторизации. Включает в себя два метода:

- Public string ControlMail

Необходим для считывания введённого пользователем текста в поле логина во время попытки авторизации, возвращает переменную типа string, соответствующую этому значению.

- Public string ControlPass

Необходим для считывания введённого пользователем текста в поле пароля во время попытки авторизации. Возвращает переменную типа string содержащую это значение.

Пример регистрации пользователя и прохождение авторизации, с выводом всех промежуточных значений, представлен на рисунке 16.

На рисунке 17 приведён пример добавления пользовательских логина и пароля в файл базы данных.

```
What do you want to do:
1 - Save login and password
2 - Generate password
3 - Update login and password
4 - View login and password
5 - Delete login and password
1
Input servicename
Service34
Input login
Login34
Input password
Pas34

Ecnrypted iv  mail/pass is vAWk4a(e9~aYr.`z

Ecnrypted key mail/pass is x^Gjthos-3Iq6A#=
Datas are saved
```

Рисунок 16 – Пример сохранения введённых пользователем логина и пароля от сервиса

На рисунке 18 продемонстрирован файл базы данных UserData, с добавленными с рисунка 17 логином и паролем в зашифрованном виде.

	Id	ServiceName	Login	Password	CardNumber	CardCVV	Description
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	Steam	Q5y3Cf++QKDsxpFRdaWK+g==	1kpLGiU+6RqEEHA30LEuUw==	NULL	NULL	NULL
2	2	Vk	biba2	biba2	NULL	NULL	NULL
3	3	steam	nUAq75aD4zVKN9hPRyyP2A==	1kpLGiU+6RqEEHA30LEuUw==	NULL	NULL	NULL
4	4	Steam	BRJDSVVR8Brz1PmQKURWmQ==	FU1dXPzzjuLRLGfNDruCGA==	NULL	NULL	NULL
5	5	Vk	biba2	biba2	NULL	NULL	NULL
6	6	biba	1VjZ+sN7i1zTwByu5sFoWw==	FU1dXPzzjuLRLGfNDruCGA==	NULL	NULL	NULL
7	7	Steam	1xQqeJZdOEGQm1LzwS5C3g==	PJ9fKwlyehd5YhuM91MhlA==	NULL	NULL	NULL
8	8	Vk	biba2	biba2	NULL	NULL	NULL
9	9	Steam	JmOMLi1VUyZZA3FgIIF7BQ==	Ko9T21xNSXCnK/Ca2iEJpQ==	NULL	NULL	NULL
10	10	Vk	biba2	biba2	NULL	NULL	NULL
11	11	Steam	0SbxNnpw5VdVoS3fG0l5Kg==	1HZU3WhWjZ0FclfvhwfQ7A==	NULL	NULL	NULL
12	12	Vk	biba2	biba2	NULL	NULL	NULL
13	13	Steam	oIsxmH6pmiovNa6BFfiVAQ==	6UTvduRMGSzIjC5PY6ozhQ==	NULL	NULL	NULL
14	14	Vk	biba2	biba2	NULL	NULL	NULL
15	15	Steam	hpUjWJOkH8rGJJfSP2n2Ag==	CWRN26nSrifjGlhjCSMxbA==	NULL	NULL	NULL
16	16	Vk	biba2	biba2	NULL	NULL	NULL
17	17	Service34	mBNWKJRziq/BBggG0oo11w==	CWRN26nSrifjGlhjCSMxbA==	NULL	NULL	NULL

Рисунок 17 – Файл, хранящий записанным все пользовательские пароли

На рисунке 19 в 11 строчке можно заметить ключ шифрования и вектор инициализации сохранённых пользовательских данных от сервиса Service 34.

	Id	ServiceName	EncKey	EncIV
	Фи...	Фильтр	Фильтр	Фильтр
1	1	Steam	A4Y}qF}Dt]3}tpT~	g#XS7P]Ik6-JqKf6
2	2	Registration	!%=3-mdl+MnSEm+X	QOx]x}B~a!]ZmRJy
3	3	steam	&QE]R-5HcU=-*epB	~rob$%sp,((zJtbK
4	4	Steam	sbkL7hnijXm2]jZ`	{9V7RUMCh1mNHzyB
5	5	biba	-3zsmbwfRb^g7k[5	xGxo^`[t3AV8!Xy%
6	6	Steam	lQ]Mk-HGQhw3,&3Q	i!991^]nJI2Nr-}*
7	7	Steam	,x`36pnk`L@=E9ce	hr*v^14ZF*hFuo&a
8	8	Steam	fjF#p@iyXX&VD_ra	zXnxZOMoP~mSN6Vq
9	9	Steam	{buNVl0bM{wB{%zd	7xxB22)fdh0nIOJ,
10	10	Steam	iO5T_8mcPaNphkv(	3Z=2Qspp,(Ypm7S!
11	11	Service34	x^Gjthos-3Iq6A#=	vAWk4a(e9~aYr.`z

Рисунок 18 – Файл, хранящий ключ шифрования и вектор инициализации добавленных данных

Таким образом в данном разделе было приведено краткое описание средств реализации, описана архитектура разрабатываемого программного обеспечения и показана структурная зависимость в базе данных, а также дано её объяснение. Затем были продемонстрированы некоторые из реализованных модулей, предоставлено их подробное описание и приведены примеры использования.

В результате выполнения проектной части данной работы было спроектировано приложение, обеспечивающее управление пользовательскими данными, а также хранение их в зашифрованном виде при помощи алгоритма

AES-256. Произведённое программное обеспечение хранит данные пользователя в нескольких файлах базы данных, то есть пользователь, отправляя запросы с клиентской части приложения и, обращаясь к серверу с каким-то одним запросом, вынуждено вынужденно взаимодействует и с другими файлами, не замечая этого. Как пример, отправляя запрос на получение хранящихся в базе данных логина и пароля, пользователь обращается не только к файлу Datapss, в котором находятся эти данные, но также отправляет автоматически генерируемый запрос к файлу, хранящему ключи дешифровки.

4. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

4.1 Введение в экономическое обоснование

После приведённого анализа в первом разделе данной выпускной квалификационной работы, можно с уверенностью сказать, что разработки в данной области являются актуальной темой.

Разработанный проект принципиально отличается от аналогичных ему приложений открытым исходным кодом, что с экономической точки зрения выгоднее, так как это уменьшает затраты на отдел тестирования.

Помимо этого, в приложении предусмотрен пробный месячный период премиальной версии, а само приложение при этом является бесплатным.

Для выполнения экономического обоснования необходимо выполнить следующие расчёты:

- Составить план-график выполнения работ, позволяющий определить совокупную трудоёмкость проделанной разработки;
- Провести оценку величины заработной платы и социальных отчислений, для лиц выполнявших работу;
- Провести оценку затрат на покупку необходимых материалов;
- Оценить затраты на услуги сторонних организаций;
- Рассчитать затраты на содержание и эксплуатацию оборудования, участвующего в разработке;
- Вычислить величину амортизационных отчислений, используемых для разработки основных средств;
- Рассчитать затраты на используемое ПО;
- Оценить сопутствующие расходы;
- Рассчитать совокупную величину затрат, связанных с разработкой продукта.

Для выполнения расчётов использовалось учебно-методическое пособие для дополнительного раздела [7].

4.2 Календарный план выполнения работы

Перед начал расчёта затрат на разработку программного продукта, необходимо составить детализированный план, отразив все моменты проектирования. Под проектированием понимается совокупность работ, необходимые к выполнению для достижения поставленных на ВКР целей.

Основой для детализированного плана будет календарный план работы над ВКР.

Для расчёта времени, затраченного на проектирование, необходимо определить продолжительность работы над каждым пунктом плана. Продолжительность определяется либо по нормативам, либо по факту, либо с помощью экспертных оценок по формуле 1:

$$t_j^o = \frac{3t_{min}+2t_{max}}{5}, \quad (1)$$

где t_j^o - ожидаемая длительность работы с индексом j, t_{min} и t_{max} наименьшая и наибольшая по мнения эксперта длительность работы [7].

Расчёты приведены в таблице 2:

Таблица 2 – Продолжительность работы

№	Наименование работы	Длительность работы, ед.t		
		t_{min}	t_{max}	t_0
1	Разработка ТЗ	4	7	5,2
2	Анализ предметной области и работа с литературой	10	12	10,8
3	Разработка архитектуры ПО	8	10	8,8
4	Разработка продукта по предложенной архитектуре	10	15	12
5	Оформление пояснительной записки	7	11	8,6
ИТОГО		39	55	54

По итогам расчётов получены следующие данные о продолжительности работ:

- Общая минимальная продолжительность ($t_{min)}$ составляет 39 дней;
- Общая максимальная продолжительность ($t_{max)}$ составляет 55 дней;
- Общая ожидаемая продолжительность составляет 54 дня.

Затем для каждого исполнителя необходимо определить ставку заработной платы за единицу времени (в нашем случае за день).

Руководителем выпускной квалификационной работы является д.т.н. кафедры САПР Горячев А.В., согласно приказу ректора № ОД/0432 от 15.10.2020 «Об индексации оплаты труда», месячный оклад руководителя составляет 49000 рублей. Для студента в качестве месячной заработной платы принимается плата инженера. Расчёты проводятся на основании предположения о 21 дневном рабочем месяце [7].

Таким образом, ставка руководителя составляет:

$$ЗП_{рук} = \frac{49000}{21} = 2333{,}3 \text{ руб/день}$$

Ставка студента рассчитывается следующим образом:

$$ЗП_{студ} = \frac{10900}{21} = 519 \text{ руб/день}$$

Результаты проведённых расчётов представлены в таблице 3.

Таблица 3 – Значения трудоёмкости и ставки исполнителей (Начало)

№	Этапы и содержание выполняемых работ	Исполнитель	Трудоемкость., t_0, ед. t	Ставка, руб./ед. t
1	Разработка ТЗ	Руководитель	5,2	2333,3
2	Анализ предметной области и работа с литературой	Студент	10,8	519
3	Разработка архитектуры ПО	Студент	8,8	519
4	Разработка продукта по предложенной архитектуре	Студент	12	519

Таблица 3 – Значения трудоёмкости и ставки исполнителей (Окончание)

5	Оформление пояснительной записки	Студент	8,6	519

4.3 Расчёт себестоимости разработки продукта

На основе данных из предыдущего пункта о трудоёмкости и рабочей ставки определяются расходы на основную заработную плату исполнителей. Расчёты производятся по формуле 2:

$$З_{\text{осн.з./пл}} = \sum_{i=1}^{k} T_i * C_i, \quad (2)$$

где $З_{\text{осн.з./пл}}$ представляют собой расходы на основную заработную плату исполнителей, k – количество исполнителей, T_i – время, затраченное исполнителем на проведение исследования, C_i – ставка исполнителя.

Расходы на дополнительную заработную плату определяются по формуле 3:

$$З_{\text{доп.з./пл}} = З_{\text{осн.з./пл}} * \frac{H_{\text{доп}}}{100}, \quad (3)$$

где $З_{\text{доп.з./пл}}$- представляют собой расходы на дополнительную заработную плату исполнителей, $З_{\text{осн.з./пл}}$ представляют расходы на основную заработную плату исполнителей, $Н_{\text{доп}}$ – норматив дополнительной заработной платы (%). При выполнении данной работы норматив принимается за значение около 12-14%.

Таким образом, основная заработная плата руководителя и студента равны соответственно:

$$З_{\text{осн.з./пл(рук)}} = 5{,}2 * 2333{,}3 = 12133{,}2 \text{ руб}$$
$$З_{\text{осн.з./пл(студ)}} = 40{,}2 * 519 = 20863{,}8 \text{руб}$$

Дополнительная заработная плата руководителя и студента тогда равны:

$$З_{\text{доп.з./пл(рук)}} = 12133{,}2 \cdot \frac{13}{100} = 1577{,}31 \text{ руб}$$

$$З_{доп.з./пл(студ)} = 20863{,}8 \cdot \frac{13}{100} = \ 2712{,}29 \text{ руб}$$

Отчисления на социальные нужды связаны с дополнительной и основной заработной платой и вычисляются по формуле 4:

$$З_{соц} = (З_{осн.з./пл} + З_{доп.з./пл}) * \frac{H_{соц}}{100}, \quad (4)$$

где $З_{соц}$ – отчисления на социальные нужды с заработной платы, $З_{осн.з./пл}$ – основная заработная плата исполнителя, $З_{доп.з./пл}$ – дополнительная заработная плата исполнителя, $H_{соц}$ – тариф страховых взносов на обязательное социальное, пенсионное и медицинское страхование (%), его значение составляет 30% [7].

Таким образом отчисления на социальные нужды для руководителя и студента равны:

$$З_{соц(рук)} = (12133{,}2 + 1577{,}31) * \frac{30}{100} = 4973{,}06 \text{ руб}$$

$$З_{соц(студ)} = (20863{,}8 + 2712{,}29\ ) * \frac{30}{100} = 7072{,}82 \text{ руб}$$

При разработке программного продукта необходимо провести оценку затраченных в процессе работы материалов. Их расчёт производится по формуле 5:

$$З_{М} = \sum_{l=1}^{L} G_l Ц_l \left(1 + \frac{H_{т.з}}{100}\right), \quad (5)$$

где $З_{М}$ – затраты на материалы (руб.); l – индекс вида материала; G_l – норма расхода l-того материала на единицу продукции (ед.); $Ц_l$ – цена приобретения единицы l-го материала (руб./ед.); $H_{т.з}$ – норма транспортно-заготовительных расходов (%).

Для обсуждения задания на выпускную квалификационную работу была приобретена тетрадь размером А4 и две ручки синего цвета.

Расчёт затрат на израсходованные материалы представлены в таблице 4.

Таблица 4 – Перечень материалов

№ п/п	Материал	Норма расхода, ед.	Цена за единицу, р.	Сумма, р.
1	Тетрадь с листами размером А4	1	150	150
2	Ручка синего цвета	2	17	34
			Итого:	184
			ТЗР	10%
			Итого с учетом ТЗР	222,4

Общая сумма затрат на материалы составила 222,4 рублей, расчёты были проведены по формуле %%.

Во время разработки программного продукта использовались услуги сети Интернет. Поскольку услуги сторонних организаций учитываются по их фактической стоимости за вычетом НДС. Для услуги доступа к сети установлена НДС в 20% [7]. Расчет НДС производится по формуле:

Сумма НДС = Стоимость услуг * ставка НДС / (100 + ставка НДС)

Стоимость услуг без НДС = Стоимость услуг – Сумма НДС

Таким образом по тарифу на сеть Интернет, стоимостью 850р, сумма ндс составляет:

Сумма НДС = 850 *20 / (100+20) = 141.6 рублей.

Тогда фактическая стоимость услуг = 850-141.6 = 708.4 рубля.

Также затраты на содержание и эксплуатацию оборудования включают затраты на электроэнергию, потребляемую во время работы. Эти затраты определятся из расчёта на 1 час работы оборудования с учётом стоимости и производительности оборудования. Расчёты производятся по формуле 6:

$$З_{эо} = \sum_{i=1}^{m} C_i^{мч} * t_i^{м}, \quad (6)$$

где $З_{эо}$ – затраты на содержание и эксплуатацию оборудования, $C_i^{мч}$- расчётная себестоимость одного машино-часа работы оборудования, $t_i^{м}$- количество машино-часов, затрачиваемых на выполнение технологической операции.

Допустим, что расчётная стоимость одного машина-часа работы оборудования равна 6 рублей, таким образом, затраты на эксплуатацию персонального компьютера, то есть расходы на электроэнергию равны 2 388 рублей.

Дополнительно во время разработки был использован текстовый редактор Microsoft Word 2016, стоимость которого также необходимо добавить пропорционально количеству работы в нём. Год подписки тарифного плана равен 4399 рублей. Значит фактические затраты во время выполнения работы на него равны 544,9 рублей.

Расчёт амортизационные отчисления по основному средству i за год вычисляются по следующей формуле 7:

$$A_i = Ц_{п.н.i} * \frac{H_{ai}}{100}, \quad (7)$$

где A_i– амортизационные отчисления за год по *i*-му основному средству, $Ц_{п.н.i.}$ – первоначальная стоимость *i*-го основного средства, H_{ai} – годовая норма амортизации *i*-го основного средства (%).

Проведём расчёты годовых амортизационных отчислений по всех техническим средствам, приняв годовую норму амортизации за 33%.

Технические материальные средства включают в себя персональный компьютер стоимостью 89900 рублей, монитор – 8990 и комплект аксессуаров за 9690 рублей, а также принтер за 9390. Тогда амортизационные отчисления равны:

$$A = (89900+8990+9690+9390) * \frac{33}{100} = 38930{,}1 \text{ рублей.}$$

Рассчитаем величину амортизационных отчислений по i-му основному средству, используемому во время выполнения выпускной работы по формуле 8:

$$A_{i\text{ВКР}} = A_i * \frac{T_{i\text{ВКР}}}{12}, \quad (8)$$

где $A_{i\text{ВКР}}$ – амортизационные отчисления по -му основному средству, используемому студентом в работе над ВКР, A_i – амортизационные отчисления за год по i-му основному средству, $T_{i\text{ВКР}}$ – время, в течение которого студент использует i-ое основное средство (мес.).

Время выполнения вкр составляет 2 месяца, произведя расчёты по формуле амортизационные отчисления по основным материальным средствам составили 6834, 61 рублей [7].

Теперь необходимо определить величину накладных расходов. В качестве прямых расходов будут определена сумма затрат на разработку ПО, эксплуатацию оборудования и используемых программ. В качестве косвенных расходов будут определены затраты на аренду помещения для разработчиков и эта величина равна 32000 рублей.

Таким образом, прямые расходы на выполнение работы составляют 153683, 31 рублей.

Косвенные расходы составляют 32000 рублей.

Процент накладных расходов составляет 21% и определяется по формуле отношения косвенных расходов к прямым.

Рассчитаем величину накладных расходов на разработку программного продукта, как произведение прямых расходов на процент накладных, она составит 32273,49 рублей [7].

Для расчёта совокупной величины затрат на разработку программного продукта, отразим все полученные величины в таблице 5:

Таблица 5 – Расчёт совокупной величины затрат

№ п/п	Наименование статьи	Сумма, руб.
1	Расходы на оплату труда	32 997
2	Отчисления на социальные нужды	11 922,72
3	Материалы	222,4
4	Расходы на содержание и эксплуатацию оборудования	12 045,88
5	Амортизационные отчисления	38930,1
6	Программное обеспечение	544,9
7	Спецоборудование	32273,49
8	Затраты по работам, выполняемым сторонними организациями	0
ИТОГО затрат		128936,49

Таким образом в заключении организационно-экономического раздела данной работы можно отметить, что разработанный программный продукт потребовал относительно небольших затрат, а в перспективе является коммерчески успешным проектом. Также данный продукт имеет множество возможностей к последующей доработке [7].

С точки зрения социального эффекта, разработанный продукт оказывает влияние на образ жизни людей, помогая им хранить пароли от учётных записей, использующихся в повседневной жизни удобным и надёжным способом.

С точки зрения научно-технического эффекта, разработанный продукт оказывает влияние на рост получаемой информации. Это обосновывается тем, что у продукта открытый исходный код, поэтому каждый пользователь может ознакомиться с тем, как работает программа изнутри. Это увеличивает уровень доверия пользователей к продукту.

С точки зрения экономического эффекта, разработанный продукт окажет влияние на цену остальных продуктов, так как наше приложение является бесплатным, из-за чего многие пользователи предпочтут наш продукт аналогам.

А премиальная версия продукта существенно дешевле премиальных версий аналогичных продуктов.

Также, в данном разделе были проведены расчёты затрат на разработку продукта и подготовку, и защиту дипломной работы, которые составили 128936,49 рублей.

ЗАКЛЮЧЕНИЕ

В данной дипломной работе была произведена разработка серверной части приложения для хранения паролей. Разработка производилась с учётом современных требований пользователей, выявленных во время анализа существующих программных средств для решения задачи надёжного хранения пользовательских паролей.

В качестве алгоритма шифрования был выбран один из самых популярных и надёжных алгоритмов AES-256. Также во время теоретического исследования актуальных алгоритмов криптографической защиты были рассмотрены другие алгоритмы, которые в перспективе продолжения развития приложения могут быть подключены. Во время исследования современных методов защиты данных были выделены такие методы, как двухфакторная аутентификация и использование комбинированных алгоритмов шифрования, эти методы также могут быть реализованы при последующем развитии данного продукта.

Кроме этого, функционал приложения может быть также дополнен при последующей поддержке развития, особенного внимания заслуживают такие функции, как одновременный многопользовательский доступ и функция автозаполнения форм авторизации в сервисах, пароли к которым хранятся в приложении

В последнем разделе данной работы, именуемым организационно-экономическим, были проведены расчёты денежных и временных затрат на разработку продукта, благодаря которому можно сделать вывод об относительно низких затратах на разработку продукта. Общие затраты на разработку и подготовку к защите составили 128936,49 рублей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальный сайт приложения LastPass // https://www.lastpass.com //
2. Официальный сайт приложения 1Password // https://1password.com //
3. Официальный сайт приложения Dashlane // https://www.dashlane.com //
4. Официальный сайт приложения Bitwarden // https://bitwarden.com //
5. Официальный сайт приложения MultiPassword // https://multipassword.com/ //
6. Сайт-сборник современных алгоритмов шифрования // https://www.cryptool.org/ //
7. Лебедева Т.Н. Выполнение дополнительного раздела ВКР «Экономическое обоснование» специалистами СБГЭТУ «ЛЭТИ»: учеб.-метод. пособие. СПб.: Издательство СПБГЭТУ «ЛЭТИ», 2022 – 36с.

ПРИЛОЖЕНИЕ А ДОПОЛНИТЕЛЬНЫЕ МАТЕРИАЛЫ

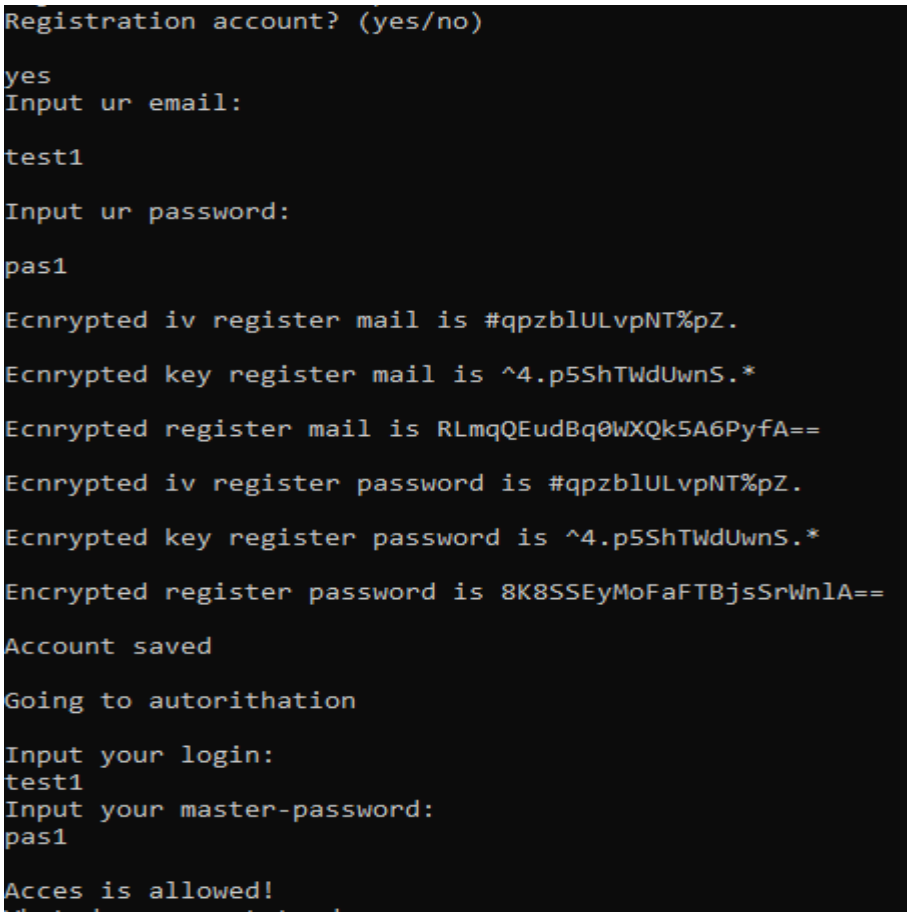

Рисунок 19 – Пример регистрации и авторизации пользователя в приложение

ПРИЛОЖЕНИИ Б ССЫЛКА НА РЕПОЗИТОРИЙ GITHUB

Код разработанного продукта находится в репозитории GitHub и доступен по следующей ссылке:

https://github.com/zhenya-ushin/app4pas