|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ ИВАНОВСКОЙ ОБЛАСТИ  Областное государственное бюджетное профессиональное образовательное учреждение  «Ивановский промышленно-экономический колледж» | | | | | | |
|  | | | | | | |
| **КУРСОВОЙ ПРОЕКТ** | | | | | | |
|  | | | | | | |
| **Разработка мобильного приложения для генерации паролей и их безопасного хранения** | | | | | | |
|  | | | | | | |
| **ИВПЭК. 09.02.07. 2** | | | | | | |
|  | | | | | | |
|  | | **Специальность:** | | | **09.02.07 Информационные системы и программирование,**  **базовая подготовка** | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| Руководитель курсового проекта | | |  | | | А.С. Горелов |
| Выполнил обучающийся группы 407а | | | |  | | А.В. Астафьев |
|  | | | |  | |  |
|  | | | | | | |
|  | | | | | | |
| Курсовой проект выполнен и защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
|  | | | | | | |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | | | | |
|  |  | | | | | |
|  | Иваново 2024 | | | | |  |

Содержание

[Введение 4](#_Toc192144382)

[1 Концептуальное проектирование 5](#_Toc192144383)

[1.1 Теоретический вопрос 5](#_Toc192144384)

[1.2 Спецификация требований программного обеспечения 7](#_Toc192144385)

[2 Техно-рабочий проект 10](#_Toc192144386)

[2.1 Обоснование выбора средств разработки 11](#_Toc192144387)

[2.2 Разработка макета приложения 12](#_Toc192144388)

[3 Рабочая документация 15](#_Toc192144389)

[3.1 Описание разработки приложения 16](#_Toc192144390)

[Заключение 23](#_Toc192144392)

[Список использованной литературы 23](#_Toc192144393)

[Приложение А 24](#_Toc192144394)

# Введение

Программирование играет ключевую роль в современном цифровом мире, предоставляя инструменты для создания решений, направленных на повышение безопасности и удобства пользователей. Одной из важнейших задач в эпоху цифровизации является защита персональных данных, что делает разработку приложений для управления паролями особенно актуальной. Такие приложения не только упрощают взаимодействие пользователя с множеством онлайн-сервисов, но и минимизируют риски, связанные с утечкой информации.

Рост числа кибератак и требований к сложности паролей обуславливает необходимость в специализированных решениях, которые совмещают генерацию надежных комбинаций и их безопасное хранение. Мобильные приложения, как наиболее доступная и удобная платформа, позволяют пользователям эффективно управлять учетными данными, обеспечивая защиту через современные методы шифрования и биометрическую аутентификацию. Разработка подобных приложений требует глубокого понимания принципов кибербезопасности, проектирования интуитивного интерфейса и реализации надежных алгоритмов.

Целью данного курсового проекта является создание мобильного приложения для генерации и безопасного хранения паролей, сочетающего функциональность, простоту использования и высокий уровень защиты данных. Основные задачи проекта включают:

* Исследование современных методов генерации паролей и технологий шифрования для обеспечения конфиденциальности данных.
* Проектирование архитектуры приложения с акцентом на безопасность и удобство взаимодействия (UI/UX-дизайн).
* Реализация модуля генерации паролей с настройкой длины, типов символов и сложности.
* Разработка защищенного хранилища с использованием алгоритмов шифрования (AES-256).

# 1 Концептуальное проектирование

1.1 Теоретический вопрос

Концептуальное проектирование является основополагающим этапом в разработке программного обеспечения, направленным на определение базовых аспектов архитектуры и функциональности будущего приложения. Оно позволяет разработчику структурировать цели и задачи проекта, а также уточнить основные требования, что закладывает прочную основу для всех дальнейших этапов разработки.

Концептуальное проектирование позволяет сформулировать общее представление о том, каким должно быть приложение, как оно будет работать, какие проблемы пользователей оно будет решать и какие ключевые функции будут его отличать. Этот процесс помогает избежать излишней сложности и неопределенности на этапе разработки, а также минимизировать риски, связанные с возможными изменениями в структуре проекта на более поздних этапах.

Концептуальное проектирование играет важную роль на ранних этапах создания приложения, так как оно позволяет:

1. Выявить и формализовать цели проекта. Этот этап позволяет четко определить основное назначение менеджера паролей, его ключевые функции и целевую аудиторию. В нашем случае, цель разработки – создание надежного и удобного приложения для безопасного хранения и управления паролями, которое обеспечит защиту конфиденциальных данных пользователей и упростит процесс управления множеством учетных записей.
2. Определить необходимую функциональность. Концептуальное проектирование помогает выявить основные функции, необходимые для реализации целей проекта. В случае с менеджером паролей это включает систему шифрования и хранения паролей, механизмы генерации надежных паролей, функции категоризации и поиска, систему резервного копирования, интеграцию с различными платформами и браузерами, механизмы синхронизации данных между устройствами.
3. Описать архитектурные подходы. Концептуальное проектирование позволяет продумать подход к организации данных и логике приложения. Это включает выбор оптимальных алгоритмов шифрования, проектирование структуры базы данных для хранения зашифрованных паролей, определение методов аутентификации пользователей, разработку механизмов защиты от несанкционированного доступа, планирование архитектуры, обеспечивающей масштабируемость и производительность.
4. Определить стратегию безопасности. Концептуальное проектирование позволяет заложить основы безопасности приложения, включая методологию защиты пользовательских данных, протоколы шифрования и передачи информации, механизмы восстановления доступа, системы мониторинга и предотвращения угроз.

В контексте данного проекта концептуальное проектирование служит основой для создания четкой структуры мобильного приложения для планирования задач с функцией уведомлений. Оно позволяет выделить основные функции, которые будут реализованы в приложении, чтобы оно стало полезным и удобным инструментом для организации задач. Эти функции формируются на основе потребностей пользователей, которые заинтересованы в простом способе управления личными делами и получении напоминаний.

Основные задачи для данного приложения включают:

1. Создание безопасной системы хранения паролей: Приложение должно обеспечивать надежное шифрование и защиту пользовательских данных. Это включает использование современных криптографических алгоритмов, защиту мастер-пароля и безопасное хранение зашифрованных данных в локальной базе данных устройства.
2. Разработка генератора надежных паролей: Необходимо реализовать функционал для создания криптостойких паролей с настраиваемыми параметрами, такими как длина пароля, использование специальных символов, цифр и букв разного регистра. Генератор должен создавать пароли, соответствующие современным требованиям безопасности.
3. Обеспечение функционала управления паролями: Пользователь должен иметь возможность добавлять, редактировать и удалять пароли, создавать категории для организации учетных записей, копировать пароли в буфер обмена с автоматической очисткой после использования.
4. Реализация удобного пользовательского интерфейса: Интерфейс должен быть интуитивно понятным и обеспечивать быстрый доступ к основным функциям. Это включает простую навигацию по сохраненным паролям, удобный поиск и фильтрацию, а также понятную систему категоризации учетных записей.

Таким образом, основные задачи охватывают как аспекты безопасности и функциональности, так и удобство использования приложения, что позволит создать надежный и практичный инструмент для управления паролями. Этот этап является основой для более детального проектирования архитектуры, интерфейса и логики приложения, обеспечивая цельность и последовательность в процессе разработки.

1.1.2 Анализ предметной области и постановка задач

Предметная область менеджера паролей охватывает сферу информационной безопасности и управления персональными данными. В современном мире, где количество онлайн-сервисов постоянно растет, существует острая потребность в надежных инструментах управления паролями. Основные задачи проектирования включают:

1. Разработку защищенной системы хранения паролей
2. Создание генератора случайных паролей с настраиваемыми параметрами
3. Реализацию удобного и безопасного механизма доступа к данным
4. Интеграцию с системными сервисами для автозаполнения форм
5. Предметная область

Создание мобильного приложения для генерации паролей и их безопасного хранения.

1. Название ИС

"SecurePass"

1. Цель использования

Целью использования системы является обеспечение пользователей надежными и безопасными паролями, а также удобное их хранение и управление ими, что позволит повысить уровень безопасности личной информации и учетных записей.

1. Анализ осуществимости ИС

Проект технически реализуем, но требует глубокой экспертизы в области кибербезопасности и разработки защищенных систем. Основные задачи включают создание удобного интерфейса, реализацию надежного шифрования данных, интеграцию с облачными сервисами и обеспечение соответствия стандартам защиты данных (GDPR, CCPA).

|  |  |  |  |
| --- | --- | --- | --- |
| Приложение | Преимущества | Недостатки | Вывод |
| LastPass | - Кроссплатформенность и облачная синхронизация. - Удобный генератор паролей с настройкой параметров. - Интеграция с браузерами и приложениями. | - Периодические уязвимости в безопасности. - Ограниченные функции в бесплатной версии (например, выбор устройства для синхронизации). | - Усилить безопасность (регулярные аудиты, открытый исходный код). - Расширить бесплатный функционал для базовых нужд. |
| **Bitwarden** | - Открытый исходный код (прозрачность безопасности). - Бесплатная версия с полным набором функций. - Поддержка двухфакторной аутентификации. | - Интерфейс менее интуитивен для новичков. - Нет встроенного VPN или мониторинга утечек. | - Упростить интерфейс для удобства новичков. - Добавить дополнительные функции (например, мониторинг данных). |
| **1Password** | - Высокий уровень безопасности (сквозное шифрование). - Семейные планы и общий доступ. - Встроенный "режим путешествия" для защиты данных. | - Нет бесплатной версии (только пробный период). - Высокая стоимость подписки. | - Ввести бесплатный тариф с ограниченным хранилищем. - Добавить гибкие тарифы для малых групп пользователей. |

Таким образом, проект должен содержать:

* Современный и удобный дизайн: интуитивно понятный интерфейс с минималистичным стилем и приятной цветовой гаммой.
* Функциональные элементы: генерация надежных паролей с настройками сложности, безопасное хранение с использованием шифрования, возможность экспорта/импорта данных.
* Персонализированный подход: адаптивные настройки генерации паролей (длина, символы) в зависимости от предпочтений пользователя.
* Поддержка дополнительных функций: использование биометрической аутентификации (отпечаток пальца, Face ID) и уведомлений о слабых или повторяющихся паролях.
* Баланс между удобством и безопасностью: простота использования для новичков с высоким уровнем защиты данных, включая оффлайн-доступ и локальное шифрование.

1. Размещение и пользователи ИС

Клиентская часть приложения хранится на устройствах пользователей и доступна через магазины приложений (Google Play, App Store). Данные, включая сгенерированные пароли и настройки, хранятся локально с использованием шифрования. Серверная часть отсутствует или используется опционально для резервного копирования (по выбору пользователя). Пользователями ИС являются:

1. Комплекс технических средств ИТ

### Компьютерная техника:

* Мощные компьютеры для разработки, тестирования и реализации шифрования.
* Тестовые устройства (смартфоны и планшеты на Android и iOS).
* Облачные серверы (опционально для резервного копирования или аналитики использования).  
   Коммуникационная техника:
* Интернет для публикации приложения в Google Play и App Store.
* Мобильные устройства для тестирования и повседневного использования.  
  Организационная техника:
* Проекторы для презентации проекта команде или заказчикам.
* Оргтехника для документирования и планирования разработки.  
   Оперативная полиграфия:
* Принтеры для печати инструкций и промо-материалов.
* ПО для дизайна (например, Adobe Photoshop, Figma) для создания интерфейса и графики приложения.

1. Системное ПО ИТ.

Клиентская часть:

* ОС Android и iOS предоставляют API для работы, локальным хранилищем и уведомлениями.  
  Средства разработки:
* Android SDK и Android Studio для разработки под Android (языки: Java или Kotlin).
* Xcode и Swift для разработки под iOS.
* Библиотеки шифрования (например, AES) для обеспечения безопасности данных.  
   Средства тестирования:
* VirtualBox или эмуляторы (Android Emulator, iOS Simulator) для тестирования на разных устройствах.
* Инструменты анализа безопасности (например, OWASP ZAP) для проверки уязвимостей.  
  Системы обновления ПО:
* Google Play Console и App Store Connect для автоматических обновлений приложения.

Основные особенности проекта "PasswordGenix":

* Интуитивно понятный и минималистичный интерфейс.
* Генерация надежных паролей с настройками (длина, символы, сложность).
* Безопасное локальное хранение с шифрованием данных.
* Поддержка биометрической аутентификации для доступа к хранилищу.
* Отсутствие обязательной подписки или рекламы.
* Возможность экспорта/импорта паролей в зашифрованном виде.
* Уведомления о слабых или повторяющихся паролях.
* Приятный дизайн и адаптивность для разных устройств.

1.1.3 Анализ аналогов и существующих решений

На рынке представлены различные решения для управления паролями, такие как LastPass, 1Password и Bitwarden. Каждое из них имеет свои преимущества и недостатки. Наше приложение будет учитывать лучшие практики существующих решений, добавляя следующие улучшения:

В отличие от существующих приложений, разрабатываемая приложение будет ориентирована на минимализм, удобство и простоту использования, с акцентом на минимализм, простоту и глубокую адаптацию под потребности пользователей. Его основная задача — превратить процесс создания и управления паролями в интуитивно понятный.

1.2 Спецификация требований программного обеспечения

Функциональные требования:

1. Генерация паролей

Интеллектуальная генерация:

* Настройка длины пароля (8-64 символа).
* Выбор типов символов (цифры, буквы верхнего/нижнего регистра, спецсимволы).
* Автоматическая оценка сложности пароля (индикатор надежности).

Контекстные подсказки:

* Рекомендации по созданию паролей для конкретных сервисов (например, банкинг, соцсети).

1. Безопасное хранение

Шифрование данных:

* Локальное хранение с использованием AES-256.
* Облачная синхронизация с end-to-end шифрованием (Firebase, AWS).

Категоризация:

* Группировка паролей по категориям (соцсети, почта, банкинг).
* Теги для быстрого поиска.

Импорт/экспорт:

* Поддержка форматов CSV, JSON.
* Автоматическое обновление устаревших паролей.

1. Автозаполнение и интеграция

* Интеграция с браузерами (Chrome, Safari) через Accessibility API.
* Поддержка автозаполнения в мобильных приложениях.

1. Система безопасности

Биометрическая аутентификация:

* Face ID, Touch ID, отпечаток пальца.

Мониторинг утечек:

* Проверка паролей на наличие в известных базах утечек (Have I Been Pwned).

Двухфакторная аутентификация (2FA):

* Генерация TOTP-кодов.

1. Управление и настройки

Адаптивная сложность:

* Автоматическое предложение усилить слабые пароли.

Интерфейс:

* Настройка уведомлений (напоминания о смене пароля).

Резервное копирование:

Ежечасное сохранение данных в зашифрованное облако.

2. Эксплуатационные требования

1. Производительность

* Генерация пароля: ≤ 0.3 сек.
* Загрузка интерфейса: ≤ 1 сек.
* Шифрование/дешифрование данных: ≤ 0.5 сек.

1. Надежность

* Защита от MITM-атак при синхронизации.
* Автоматическое восстановление данных после сбоев.
* Резервные копии в облаке с историей изменений (до 30 дней).

1. Удобство

* Поддержка экранов от 4.7 до 12 дюймов.
* Минималистичный интерфейс с быстрым доступом к генератору и хранилищу.
* Голосовые подсказки (например, при попытке использовать слабый пароль).

1. Адаптивность

* Автоматическая подстройка под разрешение экрана.
* Персонализированные рекомендации на основе истории использования.

1. Совместимость

* Android 8.0+ (Kotlin) / iOS 14+ (Swift).
* Минимальные требования: 2 ГБ ОЗУ, 50 МБ свободной памяти

1. Платформа разработки

Инструмент:

* Android Studio — официальная среда разработки для Android-приложений, предоставляющая полный набор инструментов, включая эмуляторы и средства тестирования безопасности.
* Xcode — среда разработки для iOS-приложений с поддержкой тестирования и публикации в App Store.
* Тип приложения: Нативное приложение (под Android и iOS).

1. Язык программирования

* Kotlin (для Android): Современный язык с лаконичным синтаксисом, полной поддержкой Android API, высокой производительностью и безопасностью (Null-Safe). Удобен для работы с шифрованием и локальным хранилищем.
* Swift (для iOS): Быстрый, безопасный и современный язык, оптимизированный для iOS. Поддерживает нативные API для биометрии и шифрования данных.
* Спецификация процессов

1. Основные функциональные блоки

Генерация паролей:

* Настраиваемая генерация паролей (длина, использование букв, цифр, спецсимволов).
* Визуализация сгенерированного пароля с возможностью копирования или редактирования.

Хранение паролей:

* Безопасное сохранение паролей с использованием шифрования (например, AES-256).
* Доступ к хранилищу через биометрическую аутентификацию или мастер-пароль.

Проверка безопасности:

* Анализ паролей на слабость или повторное использование.
* Уведомления о необходимости обновления паролей.

Настройки:

* Регулировка параметров генерации паролей.
* Включение/выключение биометрии, уведомлений.
* Экспорт/импорт данных в зашифрованном виде.

Работа с мультимедиа:

* Визуальные эффекты (анимация при генерации пароля).
* Звуковые сигналы (подтверждение действия).
* Тактильная обратная связь (вибрация при успешном сохранении).

1. Архитектурный подход

Приложение будет использовать архитектуру MVVM (Model-View-ViewModel) для разделения логики, упрощения тестирования и поддержки:

* Model: Бизнес-логика, шифрование данных, работа с локальной базой (SQLite).
* ViewModel: Управление состоянием данных для интерфейса.
* View: Отображение данных и обработка действий пользователя.

1. Спецификация локального хранилища (SQLite)

Таблица "User":

* id (PRIMARY KEY): Уникальный идентификатор пользователя.
* master\_password\_hash: Хэш мастер-пароля (например, SHA-256).
* date\_registered: Дата создания профиля.

Таблица "Passwords":

* password\_id (PRIMARY KEY): Уникальный идентификатор записи.
* user\_id (FOREIGN KEY): Связь с таблицей "User".
* title: Название пароля (например, "Gmail и т.д").
* encrypted\_password: Зашифрованный пароль.

Таблица "Settings":

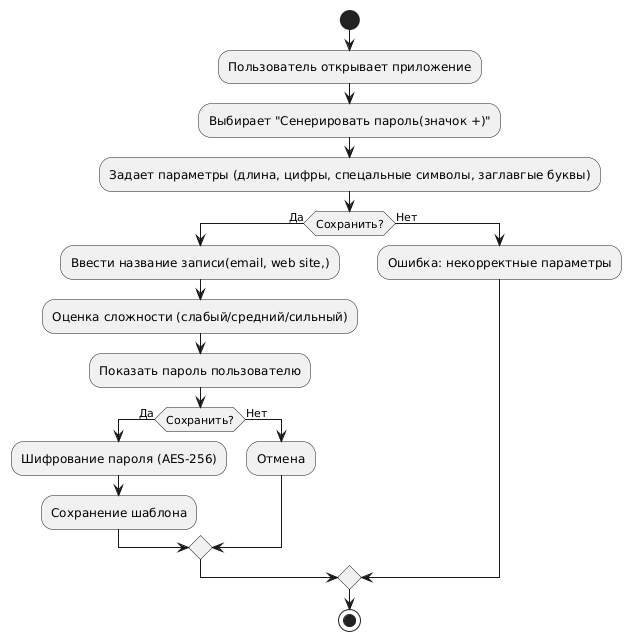
* setting\_id (PRIMARY KEY): Уникальный идентификатор настройки.
* user\_id (FOREIGN KEY): Связь с таблицей "User".
* param\_name: Название параметра (например, "password\_length").
* param\_value: Значение параметра.

1. Основные процессы

Генерация пароля:

* Пользователь задает параметры (длина, символы), приложение генерирует случайный пароль и отображает его.
* Возможность сохранить пароль в хранилище с указанием названия.

Диаграмма структуры процессов: отображает потоки данных между генерацией, настройками.



Навигационная диаграмма: показывает переходы между экранами (главный экран, генерация, хранилище, настройки).

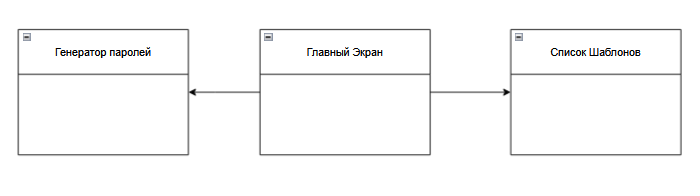
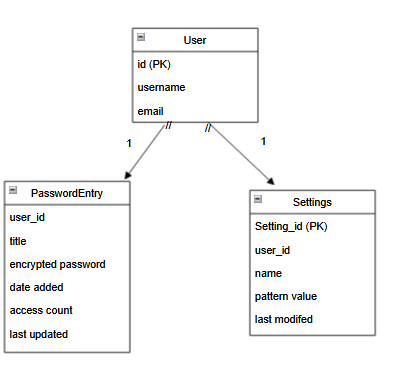


Диаграмма БД: описывает структуру таблиц и их связи.



Задание №2. Функциональная схема

1. Функциональная схема

Интерфейс пользователя (UI):

* Визуальная часть приложения: меню, список паролей, генератор, настройки.
* Отображение уведомлений и результатов анализа безопасности.

Модуль генерации паролей:

* Генерация случайных паролей на основе заданных параметров.
* Проверка уникальности и сложности пароля.

Модуль хранения:

* Шифрование и сохранение паролей в локальной базе.

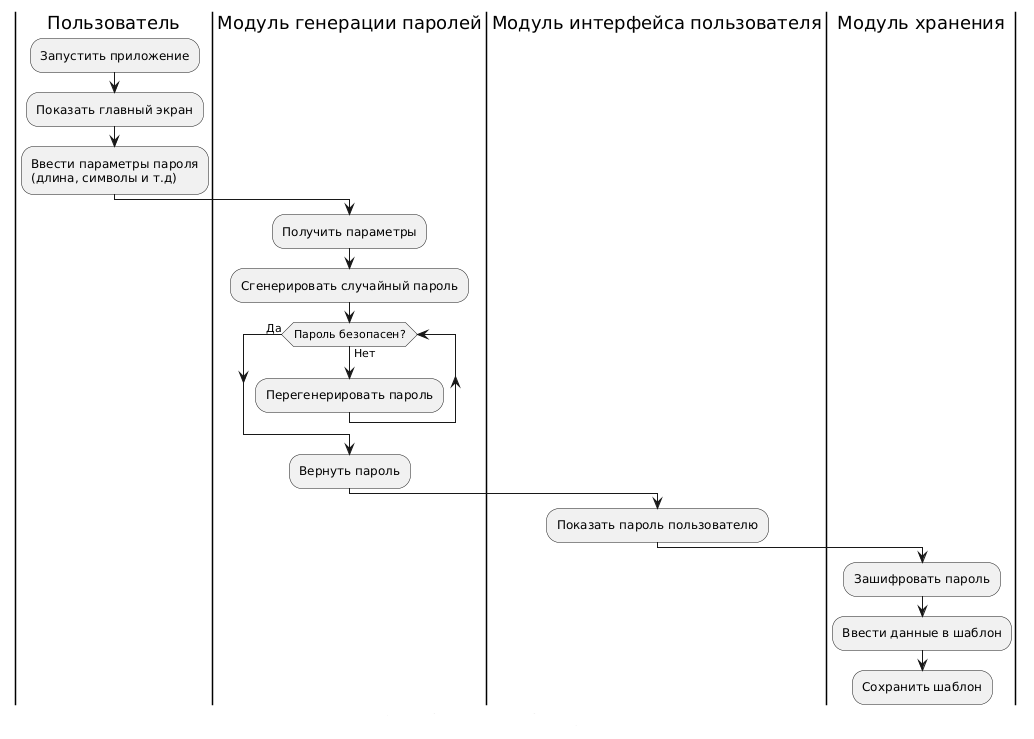
Модуль анализа безопасности:

* Проверка паролей на слабость или повторение.
* Генерация рекомендаций по улучшению паролей.

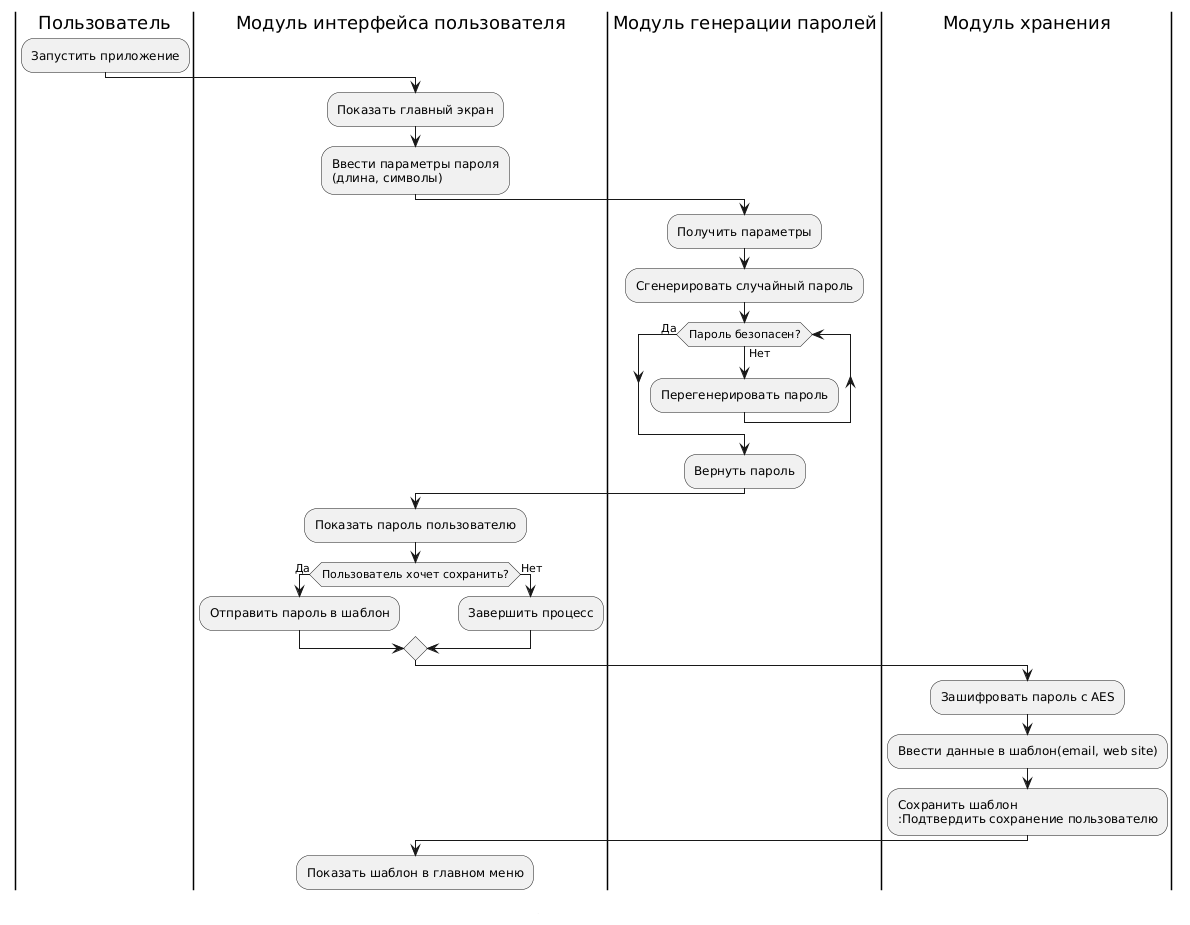
Модуль настроек:

* Регулировка параметров генерации (длина, символы).

Структурная карта Константайна – отображает иерархическую организацию экранов и модулей приложения. Она помогает визуализировать функциональную структуру и показать, как пользователь взаимодействует с системой.



Структурная карта Джексона – предназначена для отображения иерархической структуры процессов приложения, включая порядок выполнения операций. Этот подход акцентирует внимание на последовательности действий и взаимосвязях между компонентами системы.



1.2.3 Взаимодействие с концептуальным проектированием

Концептуальное проектирование менеджера паролей определяет базовые принципы безопасности и удобства использования, которые затем реализуются через функциональные и нефункциональные требования. Особое внимание уделяется балансу между безопасностью и удобством использования, что является ключевым фактором успеха приложения данного типа.

Функциональные требования, такие как система безопасного хранения паролей, механизмы генерации криптостойких паролей и возможности управления учетными данными, основываются на анализе пользовательских потребностей, выявленных в процессе концептуального проектирования. Эти требования обеспечивают реализацию основной функциональности, заложенной в начальной концепции, и служат для достижения главной цели приложения – обеспечения безопасного и удобного управления паролями пользователей.

Нефункциональные требования, включающие производительность, безопасность и удобство использования, также формируются на основе концептуального проектирования. Они определяют качественные характеристики взаимодействия с приложением и отвечают за то, чтобы пользовательский опыт соответствовал изначальной концепции – приложение должно быть не только безопасным, но и удобным в повседневном использовании.

Таким образом, концептуальное проектирование служит фундаментом для формирования требований к менеджеру паролей, обеспечивая их целостность, согласованность и направленность на создание качественного продукта. Это взаимодействие позволяет разработать приложение, которое не только соответствует современным стандартам безопасности, но и полностью удовлетворяет потребности пользователей в управлении их конфиденциальными данными.

1.3 Требование к программному средству

|  |  |  |
| --- | --- | --- |
| № | Раздел | Содержание |
| 1 | Общие сведения | 1. Наименование системы: Мобильное приложение для генерации и безопасного хранения паролей "SecurePass" (рабочее название: "SecurePass"). 2. Основание для разработки: Приложение создается для обеспечения пользователей надежным инструментом генерации сложных паролей и их безопасного хранения. Цель — повышение уровня кибербезопасности, минимизация рисков утечек данных и упрощение управления учетными записями. 3. Наименование и условное обозначение темы разработки: ⎯ Наименование: Разработка мобильного приложения "SecurePass". ⎯ Условное обозначение: "SecurePass v1.0". 4. Разработчик и заказчик: ⎯ Разработчик: Астафьев А.В. ⎯ Заказчик: Горелов А.С. 5. Плановые сроки: ⎯ Начало: 10.15.2024 ⎯ Завершение: 10.31.2024 |
| 2 | Назначение и цели | 1. Назначение: Приложение предназначено для генерации криптостойких паролей, их безопасного хранения в зашифрованном виде, автозаполнения форм авторизации и синхронизации данных между устройствами. 2. Цели разработки: ⎯Создать безопасное кроссплатформенное приложение (Android/iOS). ⎯ Реализовать генератор паролей с настройкой длины и символов. ⎯ Внедрить шифрование данных (AES-256) для локального и облачного хранения. |
| 3 | Требования к системе | 1. Требования к системе:   Генератор паролей:   * Настройка длины (8-64 символа). * Выбор типов символов (цифры, буквы, спецсимволы). Хранилище: * Шифрование данных (AES-256). * Категоризация записей (соцсети, банкинг и т.д.). * Импорт/экспорт данных (CSV, JSON). ⎯Автозаполнение: интеграция с браузерами и мобильными приложениями. ⎯Синхронизация: облачная (с end-to-end шифрованием).   Платформы: Android 8.0+, iOS 14+. ⎯ Языки: Kotlin (Android), Swift (iOS). ⎯ Библиотеки: Room/SQLite (локальное хранение), Firebase (облако).  Производительность: ⎯ Генерация пароля: ≤ 0.5 сек. ⎯ Загрузка интерфейса: ≤ 1.5 сек.  Надежность: ⎯ Резервное копирование данных ежечасно. ⎯ Защита от MITM-атак при синхронизации.  Интерфейс: ⎯ Минималистичный дизайн . ⎯ Интуитивная навигация: быстрый доступ к генератору и хранилищу. |
| 4 | Состав работ | 1. Проектирование архитектуры безопасности. 2. Разработка модуля генерации паролей. 3. Реализация зашифрованного хранилища. 4. Интеграция с биометрией и облаком. 5. Тестирование на уязвимости (OWASP Top 10). 6. Публикация в Google Play и App Store. |
| 5 | Порядок приемки | Этапы: ⎯ Аудит безопасности (проверка шифрования). ⎯ Тестирование автозаполнения и синхронизации.  Критерии: ⎯ Соответствие стандартам GDPR и PCI DSS. ⎯ Отсутствие утечек данных при тестах. |
| 6 | Требования к документации | 1. Руководство пользователя: ⎯ Инструкции по генерации паролей, настройке биометрии. 2. Документация для разработчиков: ⎯ Описание алгоритмов шифрования и API. |
| 7 | Дополнительные требования | 1. Локализация: русский, английский, немецкий. 2. Интеграция с менеджерами паролей (1Password, Last LastPass ). 3. Планы развития: усиление шифровки слабых паролей. ⎯ |

# 2 Техно-рабочий проект

2.1 Алгоритм решения поставленной задачи

1. Определение требований

Функциональные требования:

* Генерация паролей с заданными параметрами (длина, использование символов, цифр и т.д.).
* Хранение паролей в зашифрованном виде.
* Возможность создания заметок или меток для каждого пароля.
* Автоматическое заполнение паролей в браузерах или приложениях (если это необходимо).
* Возможность синхронизации данных между устройствами (если это необходимо).

Нефункциональные требования:

* Безопасность данных.
* Удобный и интуитивно понятный интерфейс.
* Поддержка различных платформ (iOS, Android).

2. Проектирование архитектуры приложения

* Выбор платформы: нативное приложение (iOS/Android) или кроссплатформенное (например, React Native, Flutter).
* Определение структуры базы данных: как будут храниться пароли и метаданные.
* Выбор алгоритмов шифрования: AES, RSA и т.д. для защиты данных.

3. Разработка пользовательского интерфейса (UI/UX)

* Создание прототипов интерфейса с использованием инструментов дизайна (Figma, Adobe XD).
* Проработка пользовательского опыта: навигация, доступ к функциям, визуальная иерархия.

4. Реализация функционала

* Генерация паролей:
* Реализация алгоритма генерации паролей с учетом заданных параметров.

Хранение паролей:

* Реализация механизма шифрования и хранения паролей в локальной базе данных (например, SQLite).

Интерфейс пользователя:

* Реализация экранов для генерации паролей, просмотра и редактирования сохраненных паролей.

5. Обеспечение безопасности

* Реализация аутентификации пользователя (например, с помощью PIN-кода, биометрии).
* Использование безопасных методов хранения и передачи данных (например, HTTPS, шифрование).

6. Тестирование

* Проведение юзабилити-тестирования для оценки удобства интерфейса.
* Функциональное тестирование для проверки всех функций приложения.
* Тестирование безопасности для выявления уязвимостей.

7. Развертывание

* Подготовка приложения к публикации в App Store и Google Play.
* Создание маркетинговых материалов (описания, скриншоты, видео).

8. Поддержка и обновление

* Сбор отзывов пользователей и исправление ошибок.
* Регулярные обновления для улучшения функционала и безопасности.

9. Документация

* Создание документации для пользователей (инструкции, FAQ).
* Подготовка технической документации для разработчиков.

## 2.2 Обоснование выбора средств разработки

Для разработки менеджера паролей я выбрал Android Studio и язык программирования Kotlin. Этот современный стек разработки обеспечивает необходимый уровень безопасности и функциональности для создания приложения, работающего с конфиденциальными данными. Рассмотрим основные преимущества выбранных инструментов:

Официальная поддержка Google:

Android Studio является официальной средой разработки для Android-приложений, что гарантирует регулярные обновления безопасности, поддержку современных криптографических библиотек и инструментов для защиты данных. Это критически важно для приложения, работающего с конфиденциальной информацией пользователей.

Стабильность и производительность:

Android Studio предоставляет встроенные инструменты для анализа безопасности кода, проверки уязвимостей и тестирования защищенности приложения. Среда разработки включает специализированные инструменты для работы с криптографическими функциями и безопасным хранением данных, что необходимо для менеджера паролей.

Kotlin – современный язык с акцентом на безопасность:

Kotlin предлагает встроенные механизмы защиты от распространенных уязвимостей, включая null-безопасность и иммутабельность данных по умолчанию. Язык обеспечивает высокую типобезопасность и предоставляет удобные инструменты для работы с криптографическими операциями, что критически важно для менеджера паролей.

Интеграция с системами безопасности::

* Android Studio обеспечивает простую интеграцию с системными механизмами безопасности Android, такими как биометрическая аутентификация, безопасное хранилище ключей и защищенное хранение данных. Это позволяет использовать встроенные механизмы защиты операционной системы.

Инструменты тестирования безопасности:

Среда разработки включает специализированные инструменты для тестирования безопасности приложения, включая анализ уязвимостей, проверку криптографических функций и тестирование механизмов защиты. Это помогает выявить и устранить потенциальные проблемы безопасности на ранних этапах разработки.

Поддержка современных стандартов безопасности:

Android Studio и Kotlin обеспечивают поддержку современных стандартов и протоколов безопасности, включая последние версии алгоритмов шифрования, безопасные методы аутентификации и защищенные протоколы передачи данных.

Выбор Android Studio и Kotlin в качестве инструментов разработки менеджера паролей обоснован их фокусом на безопасности, надежности и современных технологиях защиты данных. Эти инструменты предоставляют все необходимые средства для создания защищенного приложения, способного обеспечить безопасное хранение и управление конфиденциальной информацией пользователей.

## 2.3 Разработка макета приложения

Для проектирования макета и интерфейса приложения используется Paint.

Всего в приложении 3 экрана:

* Главное экран
* Меню создание пароля
* Экран настройки шаблона

Главное экран (Рисунок 1) содержит в себе 1 – создание пароля с шаблоном, 2 шаблон, 3 - название.

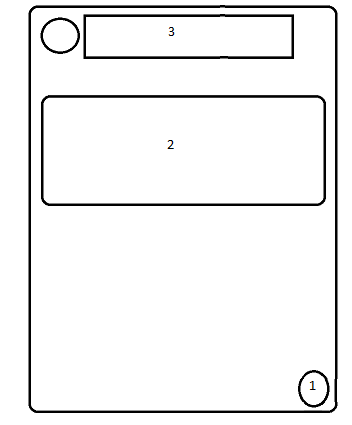


Рисунок 1 - Главный экран

Меню создание пароля (Рисунок 2) содержит 1 - Назад, 2 – Настройки генерации, 3 – Название 4 – Длина пароля 5 – Заглавные буквы 6– Строчные буквы и т.д. 7 – Сохранить.

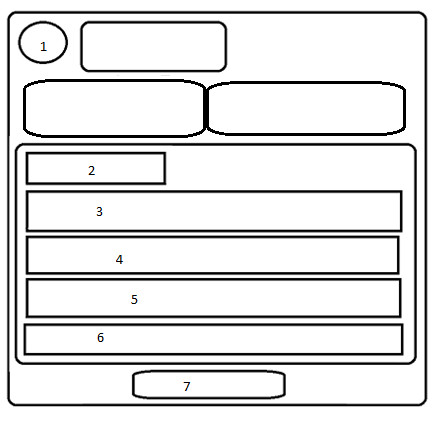
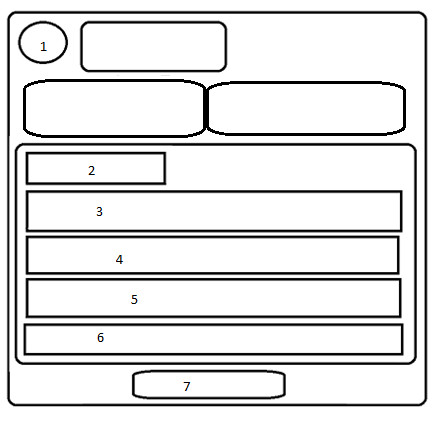


Рисунок 2 – Меню создание пароля

Экран настройки шаблона (Рисунок 3) содержит в себе 1 назад, 2 –Ввод данных, 3 Название записи, 4 Email, 5 - Пароль. 6 – Веб-сайт. 7- Сохранить.

Рисунок 3 – Экран настройки шаблона



# 3 Рабочая документация

3.1 Руководство пользователя

Диаграмма использования.

Диаграмма вариантов использования (Use Case Diagram) — это визуальное представление взаимодействия пользователей (акторов) с системой. Она определяет ключевые функции мобильного приложения "PasswordGenix" для генерации паролей и их безопасного хранения, а также связи между этими функциями и их зависимости. В данном проекте диаграмма построена с учетом ассоциаций, включения (include), расширения (extend) и наследования (generalization).

Акторы представляют собой внешние сущности, взаимодействующие с системой. В данной диаграмме он один – Пользователь (основной актор). Это конечный пользователь приложения, который хочет генерировать надежные пароли, безопасно хранить их, проверять их безопасность и настраивать приложение под свои нужды. Пользователь является центральной фигурой, так как все функции созданы для его взаимодействия с системой.

Основные функции приложения:

Функции представлены отдельными вариантами использования (Use Cases), которые являются ключевыми модулями системы, необходимыми для ее работы.

1. Генерация паролей  
   Это основной модуль приложения, позволяющий создавать надежные пароли. Без этого компонента приложение теряет свою главную ценность. Система подмодулей структурирует процесс генерации.

* Включает (include):
* Настройка параметров – пользователю необходимо задать длину, символы (буквы, цифры, спецсимволы) перед генерацией.
* Сохранение пароля – обязательный этап, где сгенерированный пароль шифруется и сохраняется в хранилище.

Расширяет (extend):

* Генерация сложных паролей – доступно только при выборе дополнительных параметров (например, увеличенная длина или уникальные символы).

1. Управление хранилищем  
   Этот вариант использования позволяет пользователю просматривать, редактировать и удалять сохраненные пароли. Он включает подмодули для полноценного управления данными.

Включает (include):

* Просмотр паролей – отображение списка сохраненных паролей в расшифрованном виде.
* Редактирование/удаление – изменение или удаление существующих записей.

1. Проверка безопасности  
   Этот модуль нужен для анализа надежности паролей и уведомления пользователя о слабых или повторяющихся комбинациях.

Включает (include):

* Анализ паролей – проверка сложности и уникальности паролей в хранилище.

1. Настройка приложения  
   Этот модуль предоставляет пользователю возможность персонализировать приложение.

Включает (include):

Экспорт/импорт данных – сохранение или загрузка паролей в зашифрованном виде.

Расширяет (extend):

Сброс данных – возможность удалить все сохраненные пароли и настройки (доступно как опция).

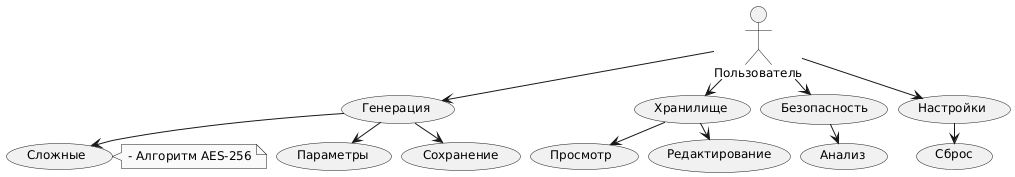


Диаграмма последовательностей.

Диаграмма последовательностей (Sequence Diagram) предназначена для отображения динамического взаимодействия между пользователем и системой в процессе работы с генерацией и хранением паролей. Данный тип диаграммы иллюстрирует порядок вызова операций, передачу сообщений между компонентами системы и этапы обработки конфиденциальных данных.

Основные цели диаграммы:

* Определение последовательности действий пользователя при генерации, сохранении и доступе к паролям.
* Демонстрация взаимодействия между модулями системы при работе с криптографическими операциями.
* Выявление ключевых этапов обработки данных (генерация, шифрование, сохранение).
* Подготовка к реализации безопасного обмена данными между компонентами приложения.

В диаграмме присутствуют следующие акторы и объекты:

|  |  |  |
| --- | --- | --- |
| Элемент | Тип | Назначение |
| Пользователь | Актор | Взаимодействует с приложением через интерфейс. |
| Главный экран | Объект | Точка входа в приложение, отображает основные разделы. |
| Генератор паролей | Объект | Создает криптостойкие пароли на основе заданных параметров. |
| Хранилище паролей | Объект | Обеспечивает безопасное шифрование и хранение учетных данных. |
| Настройки безопасности | Объект | Управление параметрами защиты (алгоритмы шифрования). |

Описание процесса:

1. Запуск приложения:

* Пользователь открывает приложение, система отображает главное меню с основными разделами
* Необходимый этап для инициализации всех компонентов безопасности

1. Генерация пароля:

* Пользователь настраивает параметры генерации (12-32 символа, комбинация символов)
* Система создает криптостойкий пароль с использованием безопасного алгоритма
* Результат автоматически подготавливается для сохранения в зашифрованном виде

1. Сохранение данных:

* Для каждой записи сохраняются:
  + Зашифрованный пароль (AES-256)
  + Метаданные аккаунта (сервис, логин)
  + Временная метка создания
* Данные сохраняются в локальное зашифрованное хранилище

1. Доступ к хранилищу:

* Просмотр списка сохраненных записей в виде:
  + Иконка сервиса
  + Замаскированный логин
  + Дата последнего использования
* Поддержка поиска по названиям сервисов

1. Настройки безопасности:

* Управление мастер-паролем
* Настройка двухфакторной аутентификации
* Параметры автоматического удаления данных
* Контроль активности устройств

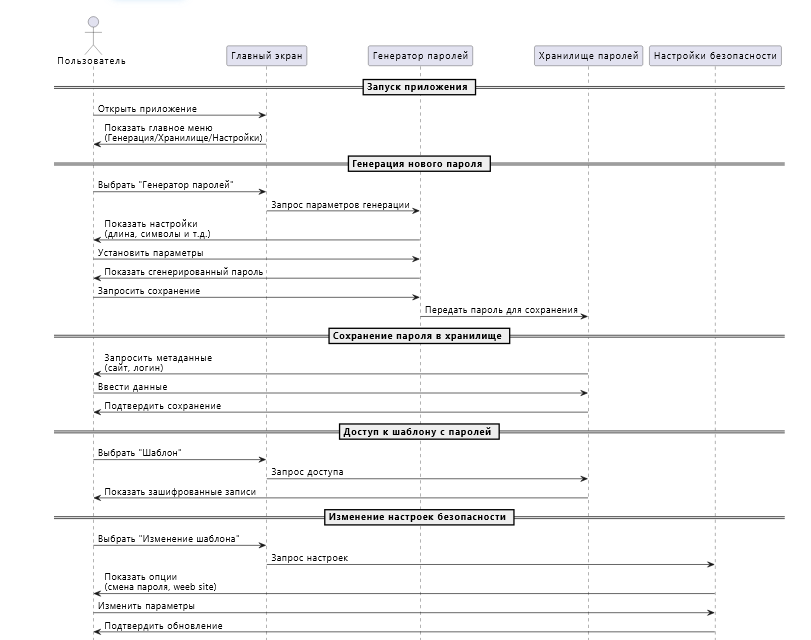


Диаграмма кооперации.

Диаграмма кооперации демонстрирует взаимодействие объектов системы в процессе генерации, сохранения и проверки безопасности паролей. Основной акцент сделан на защищенном обмене данными между компонентами приложения.

Объекты диаграммы

|  |  |
| --- | --- |
| Объект | Роль |
| Пользователь | Инициатор действий: запускает генерацию, сохраняет пароли, запрашивает доступ к данным. |
| Главный экран | Точка входа: отображает меню (генерация, хранилище, настройки). |
| Генератор паролей | Создает криптостойкие пароли на основе заданных параметров (длина, символы). |
| Хранилище паролей | Шифрует и сохраняет пароли, управляет доступом к данным (AES-256). |
| Настройки безопасности | Анализирует надежность пароля, проверяет соответствие стандартам. |

Последовательность взаимодействий

Этап 1: Запуск сценария

Пользователь выбирает в главном меню пункт «Генератор паролей».

*Роль главного экрана:* Обеспечивает навигацию между модулями приложения.

Этап 2: Генерация пароля

Главный экран передает управление Генератору паролей.

Пользователь задает параметры (длина=16, символы: A-Z, a-z, \*#%@).

*Критическая роль:* Без указания параметров генерация невозможна.

Этап 3: Проверка надежности

Генератор передает пароль объекту «Проверка безопасности».

Система анализирует пароль на соответствие критериям:

Энтропия ≥ 80 бит;

Отсутствие словарных комбинаций;

Соответствие стандарту NIST SP 800-63B.

Этап 4: Сохранение в хранилище

Проверенный пароль передается в Хранилище паролей.

Хранилище запрашивает метаданные (сервис, логин) и шифрует данные:

encrypted\_data = AES256.encrypt(password, master\_key)

Пользователь получает подтверждение сохранения.

Этап 5: Доступ к хранилищу

Пользователь запрашивает доступ к хранилищу через главное меню.

Хранилище отображает зашифрованные записи в виде:

* Сервис: Google
* Логин: joh\*\*\*@gmail.com
* Пароль: ••••••••

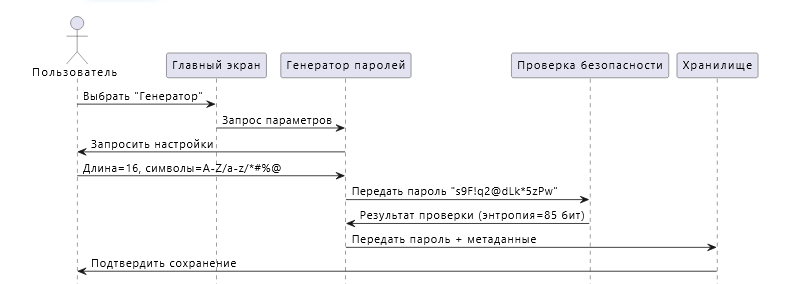


Диаграмма развертывания.

Диаграмма развертывания отображает процесс установки, настройки и начала использования приложения с акцентом на безопасность данных. Она демонстрирует, как пользователь взаимодействует с системой для защиты своих учетных записей.

Этапы развертывания

1. Начальный этап: «Пользователь заходит в магазин приложений»

Цель: Получение доверенной версии приложения.

Детали:

* Приложение должно быть загружено только из официального магазина (Google Play, App Store).
* Проверка подписи разработчика и отзывов для исключения вредоносных копий.

2. Действие: «Поиск и проверка приложения»

Цель: Убедиться в безопасности приложения перед установкой.

Детали:

* Пользователь вводит в поиск название приложения (например, «SecurePass Manager»).
* Проверяет:
* Рейтинг (не ниже 4.5/5).
* Количество скачиваний (>1 млн).
* Разрешения (не запрашивает доступ к контактам или SMS).

3. Действие: «Установка и первичная настройка безопасности»

Цель: Минимизация рисков при первом запуске.

Детали:

* Установочный пакет подписан цифровым сертификатом разработчика.
* При первом запуске пользователь обязательно настраивает:
* Мастер-пароль (12+ символов, энтропия ≥ 80 бит).

4. Действие: «Настройка параметров шифрования»

Цель: Защита локальных данных.

Детали:

* Алгоритм шифрования (по умолчанию AES-256).
* Частоту автоматического резервного копирования.

5. Действие: «Генерация первого пароля»

Цель: Проверка работоспособности системы.

Детали:

* Приложение создает пароль с параметрами:
* Длина: 16 символов.
* Символы: A-Z, a-z, 0-9, \*#&@.
* Пароль автоматически сохраняется в зашифрованном хранилище.

6. Завершающий этап: «Интеграция с аккаунтами»

Цель: Безопасное использование в реальных сценариях.

Детали:

* Пользователь привязывает приложение к сервисам:
* Google Account, Apple ID.
* Браузеры (Chrome, Safari) через защищенное API.
* Включение функции автозаполнения паролей с проверкой фишинговых сайтов.

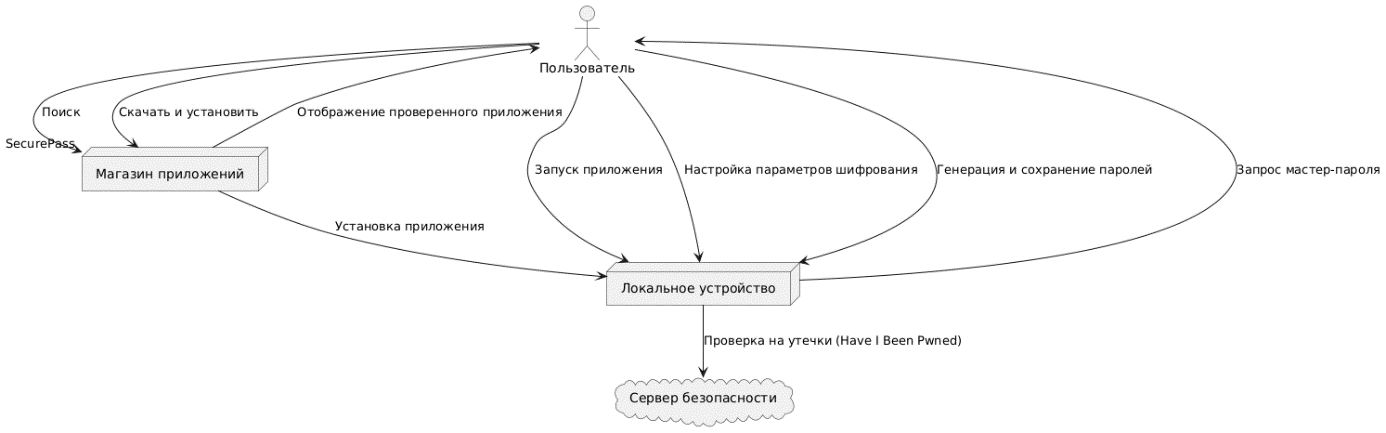


Диаграмма деятельности.

Диаграмма деятельности описывает последовательность действий пользователя при работе с генерацией, сохранением и управлением паролями. Акцент сделан на безопасность данных и взаимодействие с ключевыми компонентами системы.

1. Запуск приложения:

* Пользователь открывает приложение, инициализируются модули безопасности (проверка целостности данных).

1. Выбор генератора паролей:

* В главном меню доступны три раздела: Генератор, Хранилище, Настройки.

1. Настройка параметров:

Пользовательский режим:

* Указание длины пароля (рекомендуется 16+ символов).
* Выбор типов символов (обязательно: верхний/нижний регистр, цифры, спецсимволы).

Стандартный режим:

* Автоматическая генерация с параметрами: 16 символов, все типы символов.

1. Генерация пароля:

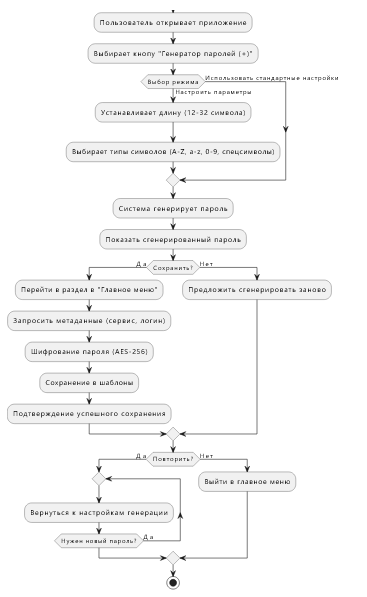
* Используется криптографически безопасный алгоритм (на основе /dev/urandom или BCrypt).

1. Сохранение в хранилище:

* Пароль шифруется мастер-ключом перед сохранением.
* Метаданные сохраняются в формате:
* "service": "Google",
* "login": "user@example.com",
* "password": "AES-256(encrypted\_data)",
* "timestamp": "2024-05-20T12:00:00Z"
* }

1. Цикл повторной генерации:

* Пользователь может повторно настроить параметры, если пароль не соответствует требованиям (например, недостаточная энтропия).



3.2 Тестирование приложения

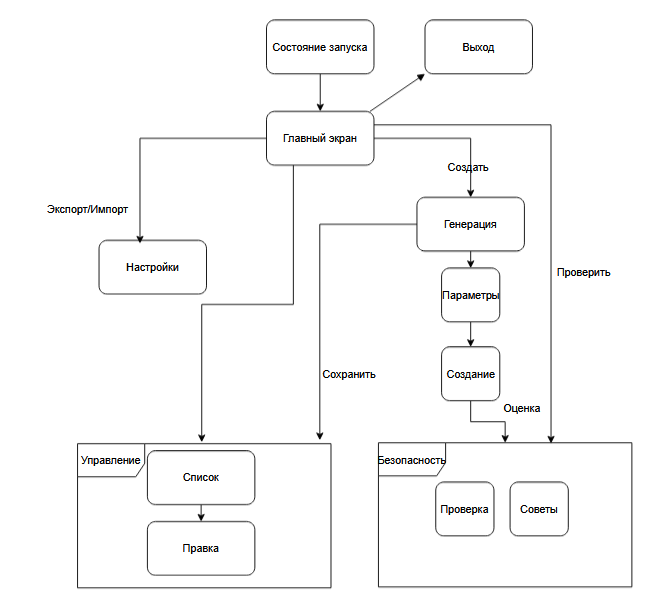
Диаграмма состояний**.**

Диаграмма состояний моделирует поведение мобильного приложения для генерации паролей в терминах различных состояний системы и переходов между ними. Она демонстрирует, как пользователь взаимодействует с системой и как приложение реагирует на его действия.Диаграмма включает следующие основные компоненты:

* Состояния – различные режимы работы приложения, в которых оно может находиться.
* Переходы – связи между состояниями, определяющие возможные изменения в ходе работы приложения.
* Начальное и конечное состояния – точки старта и завершения работы системы.

Переходы обозначают действия пользователя, которые ведут к изменению состояния системы:

* Запуск приложения –> Главный экран – приложение загружено, и пользователь видит основное меню.
* Главный экран –> Генерация – пользователь решил создать новый пароль.
* Генерация –> Параметры – пользователь настраивает параметры пароля.
* Параметры –> Создание – переход к созданию пароля с заданными параметрами.
* Главный экран –> Управление – пользователь хочет управлять сохраненными паролями.
* Управление –> Список – просмотр списка сохраненных паролей.
* Список –> Правка – редактирование выбранного пароля.
* Главный экран –> Безопасность – пользователь хочет проверить надежность паролей.
* Безопасность –> Проверка – анализ надежности выбранного пароля.
* Безопасность –> Советы – просмотр рекомендаций по созданию надежных паролей.
* Главный экран –> Настройки – пользователь решил изменить параметры приложения.
* Генерация –> Управление – сохранение созданного пароля.
* Генерация –> Безопасность – оценка надежности созданного пароля.
* Главный экран –> Выход из приложения → Конец – пользователь завершает работу с программой.



# Заключение

В рамках данного курсового проекта было разработано мобильное приложение для генерации и безопасного хранения паролей. Целью проекта было создание надежного и удобного инструмента для управления конфиденциальными данными пользователя с обеспечением высокого уровня безопасности.

В процессе разработки были выполнены следующие задачи:

* Определена концепция и архитектура приложения, включающая современные методы шифрования и безопасного хранения данных
* Разработан интуитивно понятный интерфейс пользователя, обеспечивающий удобный доступ к функциям создания, хранения и управления паролями
* Реализована система генерации криптостойких паролей с настраиваемыми параметрами сложности
* Внедрены механизмы безопасного хранения и шифрования пользовательских данных
* Реализована функция автозаполнения паролей для удобства использования
* Выполнено тестирование приложения для обеспечения надежности защиты данных и удобства использования

Проект предоставляет основу для дальнейшего развития и усовершенствования, включая добавление новых функций и оптимизацию интерфейса для лучшего взаимодействия с пользователем.

# Список использованной литературы

1. Официальная документация Android [Электронный ресурс] - https://developer.android.com/?hl=ru, 2025
2. Secure Coding Guidelines for Android. [Электронный ресурс]. URL: https://developer.android.com/topic/security/risks (дата обращения: 15.03.2024)
3. Документация Jetpack Compose [Электронный ресурс]. – URL: <https://developer.android.com/jetpack/compose>
4. Ray Wenderlich, руководства и статьи по Android Studio [Электронный ресурс] - https://www.kodeco.com/library, 2025
5. Официальная документация Kotlin [Электронный ресурс] - https://kotlinlang.org/docs/home.html, 2025
6. Руководства и гайды по Android Studio [Электронный ресурс] - https://www.linode.com/docs/guides, 2021
7. «Android Programming: The Big Nerd Ranch Guide» Билл Филипс, Крис Стюарт и Кристин Марсикано, 2016
8. Cybersecurity & Infrastructure Security Agency. Password Security Guidelines [Электронный ресурс]. – URL: https://www.cisa.gov/password-security
9. Martin R.C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Prentice Hall, 2017. – 432 p.

# Приложение А

package com.example.passwordmanager  
  
import android.os.Bundle  
import androidx.activity.ComponentActivity  
import androidx.activity.compose.setContent  
import androidx.compose.foundation.layout.fillMaxSize  
import androidx.compose.material3.MaterialTheme  
import androidx.compose.material3.Surface  
import androidx.compose.runtime.Composable  
import androidx.compose.ui.Modifier  
import androidx.navigation.NavHostController  
import androidx.navigation.NavType  
import androidx.navigation.compose.NavHost  
import androidx.navigation.compose.composable  
import androidx.navigation.compose.rememberNavController  
import androidx.navigation.navArgument  
import com.example.passwordmanager.navigation.Navigation  
import com.example.passwordmanager.screens.main.MainScreen  
import com.example.passwordmanager.screens.details.TemplateDetailsScreen  
import com.example.passwordmanager.screens.create.CreateTemplateScreen  
import com.example.passwordmanager.screens.password.PasswordDetailsScreen  
import com.example.passwordmanager.ui.theme.PasswordManagerTheme  
  
class MainActivity : ComponentActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 *setContent* **{** PasswordManagerTheme **{** Surface(  
 modifier = Modifier.*fillMaxSize*(),  
 color = MaterialTheme.colorScheme.background  
 ) **{** val navController = rememberNavController()  
 AppNavigation(navController = navController)  
 **}  
 }  
 }** }  
}  
  
@Composable  
fun AppNavigation(navController: NavHostController) {  
 NavHost(  
 navController = navController,  
 startDestination = Navigation.Main.route  
 ) **{** *composable*(route = Navigation.Main.route) **{** MainScreen(  
 onTemplateClick = **{** templateId **->** navController.navigate(Navigation.TemplateDetails.createRoute(templateId))  
 **}**,  
 onPasswordClick = **{** passwordId **->** navController.navigate(Navigation.PasswordDetails.createRoute(passwordId))  
 **}**,  
 onCreateClick = **{** navController.navigate(Navigation.CreateTemplate.route)  
 **}** )  
 **}** *composable*(  
 route = Navigation.TemplateDetails.route,  
 arguments = *listOf*(  
 *navArgument*("templateId") **{** type = NavType.StringType **}** )  
 ) **{** backStackEntry **->** val templateId = backStackEntry.arguments?.getString("templateId")  
 TemplateDetailsScreen(  
 templateId = templateId.*orEmpty*(),  
 onBackClick = **{** navController.popBackStack() **}** )  
 **}** *composable*(  
 route = Navigation.PasswordDetails.route,  
 arguments = *listOf*(  
 *navArgument*("passwordId") **{** type = NavType.StringType **}** )  
 ) **{** backStackEntry **->** val passwordId = backStackEntry.arguments?.getString("passwordId")  
 PasswordDetailsScreen(  
 passwordId = passwordId.*orEmpty*(),  
 onBackClick = **{** navController.popBackStack() **}** )  
 **}** *composable*(route = Navigation.CreateTemplate.route) **{** CreateTemplateScreen(  
 onBackClick = **{** navController.popBackStack() **}** )  
 **}  
 }**}

class MainViewModel(  
 private val preferencesService: PasswordPreferencesService,  
 private val generatorService: PasswordGeneratorService  
) : ViewModel() {  
  
 private val \_uiState = *MutableStateFlow*<MainUiState>(MainUiState.Loading)  
 val uiState: StateFlow<MainUiState> = \_uiState.*asStateFlow*()  
  
 init {  
 loadData()  
 }  
  
 private fun loadData() {  
 *viewModelScope*.*launch* {  
 *combine*(  
 preferencesService.templates,  
 preferencesService.generatedPasswords,  
 preferencesService.manualPasswords  
 ) { templates, generated, manual ->  
 MainUiState.Success(  
 templates = templates,  
 generatedPasswords = generated,  
 manualPasswords = manual  
 )  
 }.collect { state ->  
 \_uiState.value = state  
 }  
 }  
 }  
  
 fun createPasswordEntry(  
 title: String,  
 email: String,  
 password: String,  
 website: String = "",  
 notes: String = ""  
 ) {  
 val entry = PasswordData.Manual(  
 id = UUID.randomUUID().toString(),  
 title = title,  
 email = email,  
 password = password,  
 website = website,  
 notes = notes  
 )  
 preferencesService.saveManualPassword(entry)  
 }  
  
 fun getManualPassword(id: String): PasswordData.Manual? {  
 return preferencesService.getManualPassword(id)  
 }  
  
 fun deleteManualPassword(id: String) {  
 preferencesService.deleteManualPassword(id)  
 }  
  
 fun createTemplate(  
 name: String,  
 length: Int = 12,  
 useUpperCase: Boolean = true,  
 useLowerCase: Boolean = true,  
 useNumbers: Boolean = true,  
 useSpecialChars: Boolean = true  
 ) {  
 val template = PasswordTemplate(  
 id = UUID.randomUUID().toString(),  
 name = name,  
 length = length,  
 useUpperCase = useUpperCase,  
 useLowerCase = useLowerCase,  
 useNumbers = useNumbers,  
 useSpecialChars = useSpecialChars  
 )  
 preferencesService.saveTemplate(template)  
 }  
  
 fun getTemplate(id: String): PasswordTemplate? {  
 return preferencesService.getTemplate(id)  
 }  
  
 fun generatePasswordForEntry(template: PasswordTemplate): String {  
 return generatorService.generatePassword(template).password  
 }  
  
 fun generatePassword(templateId: String) {  
 val template = getTemplate(templateId) ?: return  
 val generatedPassword = generatorService.generatePassword(template)  
 val passwordData = PasswordData.Generated(  
 id = UUID.randomUUID().toString(),  
 name = template.name,  
 template = template,  
 generatedPassword = generatedPassword.password  
 )  
 preferencesService.saveGeneratedPassword(passwordData)  
 }  
  
 fun getGeneratedPassword(templateId: String): String? {  
 return preferencesService.getGeneratedPassword(templateId)?.generatedPassword  
 }  
  
 companion object {  
 private val dateFormat = SimpleDateFormat("dd.MM.yyyy HH:mm", Locale.getDefault())  
   
 fun formatDate(timestamp: Long): String {  
 return dateFormat.format(Date(timestamp))  
 }  
 }  
}  
  
sealed class MainUiState {  
 object Loading : MainUiState()  
 data class Success(  
 val templates: List<PasswordTemplate>,  
 val generatedPasswords: List<PasswordData.Generated>,  
 val manualPasswords: List<PasswordData.Manual>  
 ) : MainUiState()  
 data class Error(val message: String) : MainUiState()