МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

ОТЧЕТ

по учебной вычислительной (ознакомительной) практике

Лукьяновича Никиты Андреевича

студента 1 курса, 4 группы

специальность «Информатика»

Руководитель практики:

старший преподаватель

О.Г. Казанцева

Минск, 2025

# ОГЛАВЛЕНИЕ

[**ОГЛАВЛЕНИЕ 3**](#_heading=h.fd6iho5p21eu)

[Введение 5](#_heading=h.xceuk0w7em0t)

[Глава 1 Теоретические основы 6](#_heading=h.rzzx5zmeivon)

[1.1 Общая характеристика курса «Контроль версий с помощью Git» 6](#_heading=h.2tlmuctgw4tc)

[1.1.1 Введение в системы контроля версий и Git 6](#_heading=h.8uukox83biad)

[1.1.2 Создание репозитория и базовый рабочий процесс 6](#_heading=h.8lckyb1ue0pe)

[1.1.3 Работа с удаленными репозиториями 7](#_heading=h.9zih4ff7um9r)

[1.1.4 Просмотр истории коммитов 7](#_heading=h.ssiquvdok8h1)

[1.1.5 Ветвление и слияние 7](#_heading=h.9g22zekkqvyp)

[1.1.6 Перебазирование (Rebasing) 8](#_heading=h.lvo2usf40lg)

[1.2 Общая характеристика курса «Python» 8](#_heading=h.dxtnavva89rx)

[1.2.1 Ввод и вывод данных, операции с числами и строками, форматирование 9](#_heading=h.qsm1e57up0d)

[1.2.2. Циклы while и for 10](#_heading=h.dactard6a1g4)

[1.2.3. Коллекции данных: строки, списки, кортежи и списочные выражения 10](#_heading=h.rlpap1da33po)

[1.3 Выводы к главе 1 11](#_heading=h.5zg2pg6m2gsm)

[Глава 2 Индивидуальный проект «Emphizor» 12](#_heading=h.jxb85smod062)

[2.1 Этап предпроизводства разработки проекта «Emphizor» 12](#_heading=h.x3360p3kwigy)

[2.1.1 Концепция и Идея 12](#_heading=)

[2.1.2 Документ Концепции 12](#_heading=)

[2.1.3 Разработка Дизайн-Документа (ADD) 14](#_heading=)

[2.1.3.3 Планирование: 14](#_heading=h.hjxuym3aiecc)

[2.2 Этап производства разработки проекта «Emphizor» 15](#_heading=h.70zgtn9ww4c0)

[2.2.1.1. Создание макета интерфейса 15](#_heading=h.wlppx6zgn2hu)

[2.2.2 Программирование (Programming)  
  
 2.2.2.1. Реализация концепций (Conception Implementation) 15](#_heading=h.xfjyhrflk3dp)

[2.2.3 Оптимизация (Optimization) 16](#_heading=h.86y3a6kp45mq)

[2.2.4 План работ по этапу «Программирование» 16](#_heading=h.axycdktxl994)

[2.2.3. Дизайн Интерфйеса 17](#_heading=h.t3ay7vv5u6iv)

[2.2.4 Разработка бэкенд части приложения 18](#_heading=h.7lbrm56iji5c)

[2.2.5 Тестирование ( Testing) 18](#_heading=h.dl2qdiopsgnp)

[2.2.6 Создание Звуковых Эффектов (SFX Creation) 19](#_heading=h.2mxahf4382mb)

[2.2.7 Интеграция Звука в Приложение (Audio Integration) 19](#_heading=h.2fzj5rtwv97q)

[2.2.8 Интеграция и Тестирование (Integration & Testing) 19](#_heading=h.nnaybf56hj62)

[2.2.8.1 Альфа-Тестирование (Alpha Testing) 19](#_heading=h.oulih2qgf8au)

[2.2.8.2. Багфиксинг (Bug Fixing) 20](#_heading=h.2ru3ruasyti)

[2.2.8.3. Оптимизация (Optimization) 20](#_heading=h.z12qlm1nzhhq)

[2.3 Интерфейс приложения 20](#_heading=h.558880or4ajk)

[2.4 Архитектура проекта 30](#_heading=h.nq2mj74llo38)

[2.5 Реализация бизнес-логики приложения 31](#_heading=h.g5ckzdq0dsz7)

[2.6 Выводы к главе 2 33](#_heading=h.vwqg6gxq5n6f)

[Заключение 34](#_heading=h.9rkil56k17ed)

[Список использованных источников 35](#_heading=h.wq7u26rhw40z)

# Введение

Целью учебной вычислительной (ознакомительной) практики является закрепление полученных знаний за первый год обучения посредством выполнения специальных учебных заданий, участия в работе над командным проектом.

Задачи практики:

* Изучение инструментов для совместной разработки проектов и контроля версий.
* Подготовка и настройка программных сред и средств тестирования для прохождения тренинга по выбранному направлению.
* Знакомство с IT компанией, возможностями обучения и повышения квалификации, знакомство с разными направлениями профессиональной подготовки IT специалистов.
* Выполнение стандартных заданий.
* Выполнение заданий повышенного уровня.
* Разработка приложения «Emphizor».
* Оформление отчета.

Выбранное направление работы: разработка индивидуального проекта «Emphizor» – приложения для интервального повторения, с использованием современного и эффективного алгоритма Free Spaced Repetition Scheduler (FSRS). Данное приложение является аналогом Anki [13], Remnote [14] и других похожих. Во время работы над приложением активно использовался Git и Github с использованием модели ветвления Github Flow. Были использованы и изучены основы технологий Supabase [5] для авторизации пользователей, PostgreSQL как база данных, библиотека py-fsrs [4], pyside6 [2].

Во время УП 09.07.2025 для студентов была проведена экскурсия в IT компанию «LeverX» [7]. На этой встрече мы посетили их современный офис, послушали людей работающих здесь 4-5 лет и узнали от тимлидов чем занимаются здесь разработчики. Рассказали как попасть в компанию – на 3 курсе вуза или позже, пройти бесплатный курс по одному из направлений (back-end на [Node.js](http://node.js), ABAP, front-end и др.), и если достаточно хорошо проявить себя, компания предлагает контракт на 2 года с 3 месяцами испытательного периода.

# Глава 1 Теоретические основы

## 1.1 Общая характеристика курса «Контроль версий с помощью Git»

В процессе изучения Git использовались книга «Pro Git», а также различные онлайн-туториалы и руководства. Эта книга оказалась полезной благодаря её четкой структуре и наглядным примерам.

В рамках изучения курса «Контроль версий с помощью Git» были рассмотрены следующие основные модули и возможности, которые являются фундаментом для работы с данной системой:

### 1.1.1 Введение в системы контроля версий и Git

Модуль посвящен пониманию концепции систем контроля версий (VCS). Были рассмотрены эволюция VCS от локальных (Local VCS, например, RCS), через централизованные (CVCS, например, CVS, Subversion) к распределенным (DVCS, например, Git, Mercurial).

Особое внимание уделено преимуществам распределенных систем, таким как отсутствие единой точки отказа, полная история на каждом клиенте и гибкость в работе с удаленными репозиториями.

### 1.1.2 Создание репозитория и базовый рабочий процесс

В этом модуле изучены способы получения Git-репозитория и основные шаги работы с ним. Инициализация репозитория: создание нового репозитория в существующем проекте с помощью *git init*, что создает скрытую директорию *.git* для хранения всей истории проекта. Клонирование репозитория: Получение полной копии существующего удаленного репозитория с использованием git clone. Состояния файлов: Четкое разграничение состояний файлов в рабочем каталоге: "отслеживаемые" (tracked) и "неотслеживаемые" (untracked), а также подсостояния "неизмененные" (unmodified), "измененные" (modified) и "подготовленные" (staged) для отслеживаемых файлов.

Проверка статуса файлов: Использование команды git status для просмотра текущего состояния рабочего каталога и области подготовленных изменений.

Отслеживание новых файлов: Добавление новых или измененных файлов в область подготовленных изменений с помощью git add. Фиксация изменений (коммит): Запись подготовленных изменений в историю репозитория с помощью git commit. Изучены варианты создания коммитов с редактором сообщений или напрямую. Игнорирование файлов: Настройка файлов .gitignore для исключения определенных файлов или директорий из отслеживания Git. Просмотр изменений: Использование git diff для просмотра разницы между рабочим каталогом, областью подготовленных изменений и последним коммитом. Удаление и перемещение файлов: Удаление файлов из репозитория с помощью git rm и перемещение/переименование файлов с git mv. Отмена действий: Изучены способы отмены изменений и возврат к предыдущим состояниям с использованием git restore, git reset HEAD и git checkout --.

### 1.1.3 Работа с удаленными репозиториями

Модуль посвящен взаимодействию с удаленными репозиториями, что очень важно для командной работы: Просмотр удаленных репозиториев: Использование git remote и git remote -v для отображения списка настроенных удаленных репозиториев и их URL. Добавление удаленных репозиториев: Настройка нового удаленного репозитория с git remote add. Получение изменений (fetch и pull): Синхронизация локального репозитория с удаленным. git fetch загружает изменения без слияния, а git pull выполняет fetch и последующий merge. Отправка изменений (push): Отправка локальных коммитов в удаленный репозиторий.

Просмотр информации об удаленном репозитории: Использование git remote show для получения детальной информации о конкретном удаленном репозитории, включая отслеживаемые ветки и URL. Переименование и удаление удаленных репозиториев: Управление настроенными удаленными репозиториями с git remote rename и git remote remove.

### 1.1.4 Просмотр истории коммитов

Этот модуль охватывает мощные возможности git log для анализа истории коммитов: Базовый git log: Использование команды git log для просмотра хронологического списка коммитов с их хешами, авторами, датами и сообщениями. Параметры форматирования вывода: Применение различных опций для настройки вывода, таких как --patch (показ изменений в каждом коммите), --stat (статистика изменений файлов), --pretty (настраиваемое форматирование, например, oneline или format).

Ограничение и фильтрация вывода: Использование опций для фильтрации коммитов по дате (--since, --until), автору (--author), сообщению (--grep) или изменению содержимого (-S).

### 1.1.5 Ветвление и слияние

Один из самых мощных инструментов Git – его модель ветвления:

Принципы ветвления: Понимание того, что ветка в Git — это просто легковесный перемещаемый указатель на один из коммитов. Объяснение работы указателя HEAD.

Создание и переключение веток: Использование git branch <имя\_ветки> для создания новой ветки и git checkout <имя\_ветки> для переключения на нее. Шорткат git checkout -b <имя\_ветки> для создания и переключения одновременно.

Слияние веток: Объединение изменений из одной ветки в другую с помощью git merge. Рассмотрены типы слияний: Fast-Forward (быстрая перемотка) и Three-Way Merge (трехстороннее слияние) с созданием коммита слияния. Конфликты при слиянии: Понимание причин возникновения конфликтов и методы их разрешения (ручное редактирование файлов, использование git mergetool). Управление ветками: Просмотр списка веток (git branch), удаление веток (git branch -d), а также понимание отличий между локальными и удаленными ветками

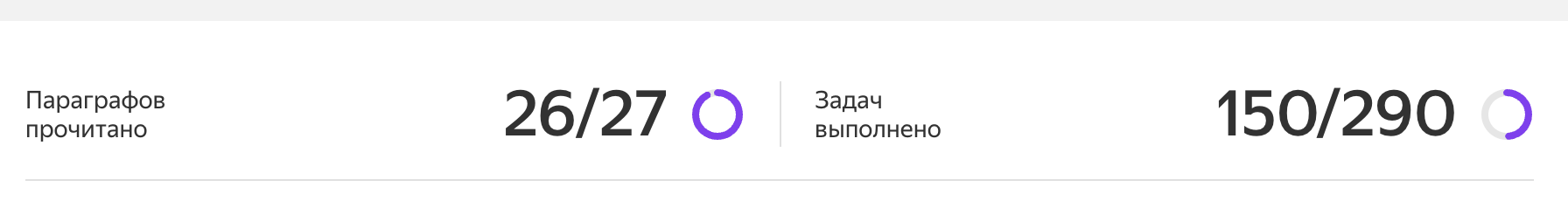
### 1.1.6 Перебазирование (Rebasing)

Данный модуль знакомит с альтернативным методом интеграции изменений. Концепция перебазирования: Изучение git rebase как способа перемещения или комбинирования последовательности коммитов на новую базовую точку. По сравнению со слиянием, перебазирование создает линейную историю проекта без коммитов слияния.

Сравнение rebase и merge: Обсуждение преимуществ (более чистая, линейная история) и недостатков (переписывание истории, проблемы с публичными коммитами) перебазирования по сравнению со слиянием. Особое внимание уделено правилу "Не перебазируйте коммиты, находящиеся за пределами вашего репозитория и на которых могли основываться другие люди", чтобы избежать проблем для других разработчиков.

## 1.2 Общая характеристика курса «Python»

Курс «Основы Python», освоенный на платформе Яндекс Образование, представляет собой фундаментальное введение в язык программирования Python. Его целью является формирование понимания основных концепций и развитие базовых навыков, необходимых для решения прикладных задач. В процессе изучения использовались материалы интерактивного хендбука "Основы Python" от Яндекс Образования.



Материалы курса структурированы с постепенным усложнением материала. Каждый раздел сопровождается подробными объяснениями, наглядными примерами кода, интерактивными квизами и практическими задачами. В отличие от множества ресурсов, "Яндекс Хендбук" не только предоставляет теорию, но и акцентирует внимание на практическом применении полученных знаний. В целом, курс и его материалы эффективные для освоения основ программирования на Python.

### 1.2.1 Ввод и вывод данных, операции с числами и строками, форматирование

В модуле изучаются основы взаимодействия программы с пользователем и работы с базовыми типами данных. Изучены функции input() для получения данных, print() для вывода, а также освоены различные способы форматирования строк, включая мощные f-строки. Рассмотрены арифметические операции и преобразование типов данных.

Пример задачи. Написать программу, которая выводит красиво оформленный чек. Все строки в чеке должны быть длиной ровно 35 символов, текст выровнен в соответствии с примером. Вводятся: название товара, цена за кг, вес товара, количество денег у пользователя. Необходимо вывести итоговую стоимость и сдачу. Код решения:

item = input()

price = int(input())

weight = int(input())

money = int(input())

total = weight \* price

change = money - total

detailed\_price = f'{weight}кг \* {price}руб/кг'

check = f'''{'Чек':=^35}

Товар:{item:>29}

Цена:{detailed\_price:>30}

Итого:{total:26}руб

Внесено:{money:24}руб

Сдача:{change:26}руб\n''' + '=' \* 35

print(check)



Пример работы программы:

****================Чек================

Товар: черешня

Цена: 3кг \* 2руб/кг

Итого: 6руб

Внесено: 10руб

Сдача: 4руб

===================================



### 1.2.2. Циклы while и for

В этом модуле были глубоко изучены основы управления повторяющимися действиями в программе. Рассмотрены циклы while для выполнения кода до тех пор, пока условие истинно, и for для итерации по последовательностям. Освоены функции range() для генерации последовательностей чисел и моржовый оператор := для более компактной записи условий в циклах.

Пример кода с моржовым оператором:

while (name := input("Введите имя: ")) != "СТОП":

print(f"Привет, {name}!")

print("Программа завершена.")



### 1.2.3. Коллекции данных: строки, списки, кортежи и списочные выражения

Данный модуль объясняет работу с упорядоченными коллекциями данных. Были изучены их общие свойства (индексация, срезы, итерация), а также специфические особенности: неизменяемость строк (str) и кортежей (tuple), изменяемость списков (list). Отдельное внимание уделено списочным выражениям (list comprehensions) как мощному и компактному инструменту для создания и преобразования списков. Пример использования списочных выражений:

numbers = [int(input()) for i in range(7)]



## 1.3 Выводы к главе 1

В рамках Главы 1 были изучены основы языка Python и системы контроля версий Git, необходимая для командной работы над проектами.

Изучены основы и получены практические навыки работы с системой контроля версий Git. Изучены ключевые концепции, такие как создание репозиториев, базовый рабочий процесс (add, commit), взаимодействие с удаленными репозиториями (fetch, pull, push), просмотр истории коммитов (git log), а также ветвление (branch), слияние (merge) и перебазирование (rebase).

Приобретены навыки программирования на языке Python, включая понимание базовых типов данных (числа, строки), работу с коллекциями (списки, кортежи), логическими операциями, циклами (for, while) и функциями. Изучено использование списочных выражений и генераторов для написания более эффективного кода.

Полученные знания по Git и программированию на Python создают теоретическую основу для выполнения проекта.

# Глава 2 Индивидуальный проект «Emphizor»

Репозиторий проекта [2].

Состав команды: Лукьянович Никита, Ткаченко Радомир.

## 2.1 Этап предпроизводства разработки проекта «Emphizor»

### 2.1.1 Концепция и Идея

Приложение аналог Anki [13] и Remnote [14] для интервального повторения, с использованием библиотеки py-fsrs.Целевая аудитория приложения - студенты, школьники, обучающиеся.

Исследование рынка: Изучили основные концепции существующих аналогов(Remnote, Anki), провели опрос целевой аудитории и выяснили, что важным фактором является высокая стоимость использования существующих аналогов.

Цели: Создание приложения дешевого, удобного для пользователей приложения

Творческая: Попробовать себя в дизайне архитектуры и интерфейса.

Техническая: Использовать pyside6 [2], supabase [5], PostgreSQL [6].

### 2.1.2 Документ Концепции

#### 2.1.2.1 Краткое описание

оконное приложение для интервального повторения, созданное наподобие существующих конкурентов, но предоставляющее пользователю выбор того, каким ИИ он желает пользоваться.

Весь функционал по созданию карт которого умещался бы на одном экране. создание карт же предусматривается как самим пользователем, так и с помощью искусственного интеллекта (генерация ответов по заданному вопросу). Карты в свою очередь должны группироваться по так называемым тегам, у одной карты может быть несколько тегов. Теги карты говорят о том, какие темы охватывает её вопрос. Для практики с картами мы предусматриваем два варианта: классический(сразу возникает вопрос и кнопка по которой можно узнать ответ и оценить своё знание темы) и Concept Connect режим(в котором на экране появляется 9 плиток, каждая из которых содержит либо вопрос, либо ответ.

Целью пользователя является совмещение плиток с соответствующими вопросами и ответами. Также в наборе есть 1 лишняя плитка, чтобы не возникло ситуации, когда в конце остаётся лишь 2 плитки, которые соответствуют друг другу).

Также мы планируем добавить в наш проект кастомизацию интерфейса, а именно пользовательский выбор цветовой темы приложения. Пользователь бы выбирал 2 цвета, которые бы и составляли все градиенты отображаемые в приложении. Всю информацию о пользователе, такую как:имя, пароль, адрес электронной почты, его теги, карты, ревью его ответов приложение будет хранить в облачном хранилище.

#### *2.1.2.2 Назначение, целевая аудитория*

наш продукт предназначен для оптимизации процесса повторения, изучения нового материала на основе современных технологий, учитывающих недавние исследования в области психологии. Предназначается же он для студентов, школьников, в целом всех людей которые занимаются саморазвитием.

#### *2.1.2.3 Основные концепции*

мэтчинг подходящих плиток в режиме Concept Connect вместе с ведением рейтинга ответа учитывающего время и правильность, квиз с отзывом в режиме Practice, выбор тегов реализован через нажатие на кнопки с нужными названиями. Учет правильности ответов пользователя, его знания карт и связь этого с частотой их появления реализована через алгоритм FSRS(Free Spaced Repetition Shcheduler)

#### *2.1.2.4 Уникальные особенности*

пользовательский выбор ИИ, кастомизация интерфейса, Concept Connect режим тренировки.

#### *2.1.2.5 Примеры похожих приложений:*

Сейчас на рынке существуют следующие аналоги: Anki, Remnote

### 2.1.3 Разработка Дизайн-Документа (ADD)

#### *2.1.3.1 Опыт использования*

создание тегов для карточек, создание карточек, группировка карт по тегам

кнопки меню, (возможно горячие клавиши)

Концепции:

Карточки: карточка — единица «знаний» в нашем приложении заключающая в себе вопрос и ответ.

Теги: карточки имеют теги, указывающие на то к какой теме они относятся.

Использование ИИ: Для удобства и экономии времени при создании карточек.

Сюжет: Вам надо выучить что-то и вы нашли наше приложение!

Элементы интерфейса: Главный экран, список тегов, экран создания карт

UI/UX: Чистый, минималистичный интерфейс , понятные переходы.

Стиль: минималистичный

Технический Дизайн: QT Designer, PySide6.

#### *2.1.3.2 Прототипирование*

Основной задачей прототипирования на данном этапе является разработка чернового варианта интерфейса, дизайна, наполнение его функционалом и далее выход к MVP и сбор обратной связи от заинтересованных людей через Telegram.

### 2.1.3.3 Планирование:

В ресурсы нашей команды входят: 2 студента, идея.

## 2.2 Этап производства разработки проекта «Emphizor»

Цель: Создание всех элементов интерфейса в соответствии с дизайн-документом и планом производства.

### 2.2.1.1. Создание макета интерфейса

Создание разметки расположения основных элементов интерфейса

#### 2.2.1.2 Наполнение макета

В соответствии с ранее разработанной разметкой расставить элементы интерфейса по своим местам

#### 2.2.1.3 Настройка стилей

Задачей этого этапа является подбор и установка стилей ранее добавленных элементов интерфейса при помощи CSS

#### 2.2.1.4 Анимация

Данный этап предполагает добавление анимаций при взаимодействии пользователя с интерфейсом. Например: анимации нажатия кнопок, анимации переходов между меню.

### 2.2.2 Программирование (Programming) 2.2.2.1. Реализация концепций (Conception Implementation)

По задумке данный этап должен в себя включать наполнение ранее созданного интерфейса функционалом для ответа приложения на действия пользователя. Также должны быть реализованы игровые режимы тренировки: Practice, Concept Connect. Помимо этого должен быть реализован пользовательский выбор цветовой темы.

#### 2.2.2.2 Разработка Искусственного Интеллекта (AI Development)

Целью этого периода разработки является разработка функционала связанного с взаимодействием пользователя с выбранным им искусственным интеллектом. А именно реализация автозаполнения поля ответа карты при заданном пользователем вопросе.

#### 2.2.2.3 Добавление SFX

Задачей этого пункта устанавливается поиск подходящих звуковых эффектов для приложения с их последующей интеграцией в продукт.

### 2.2.3 Оптимизация (Optimization)

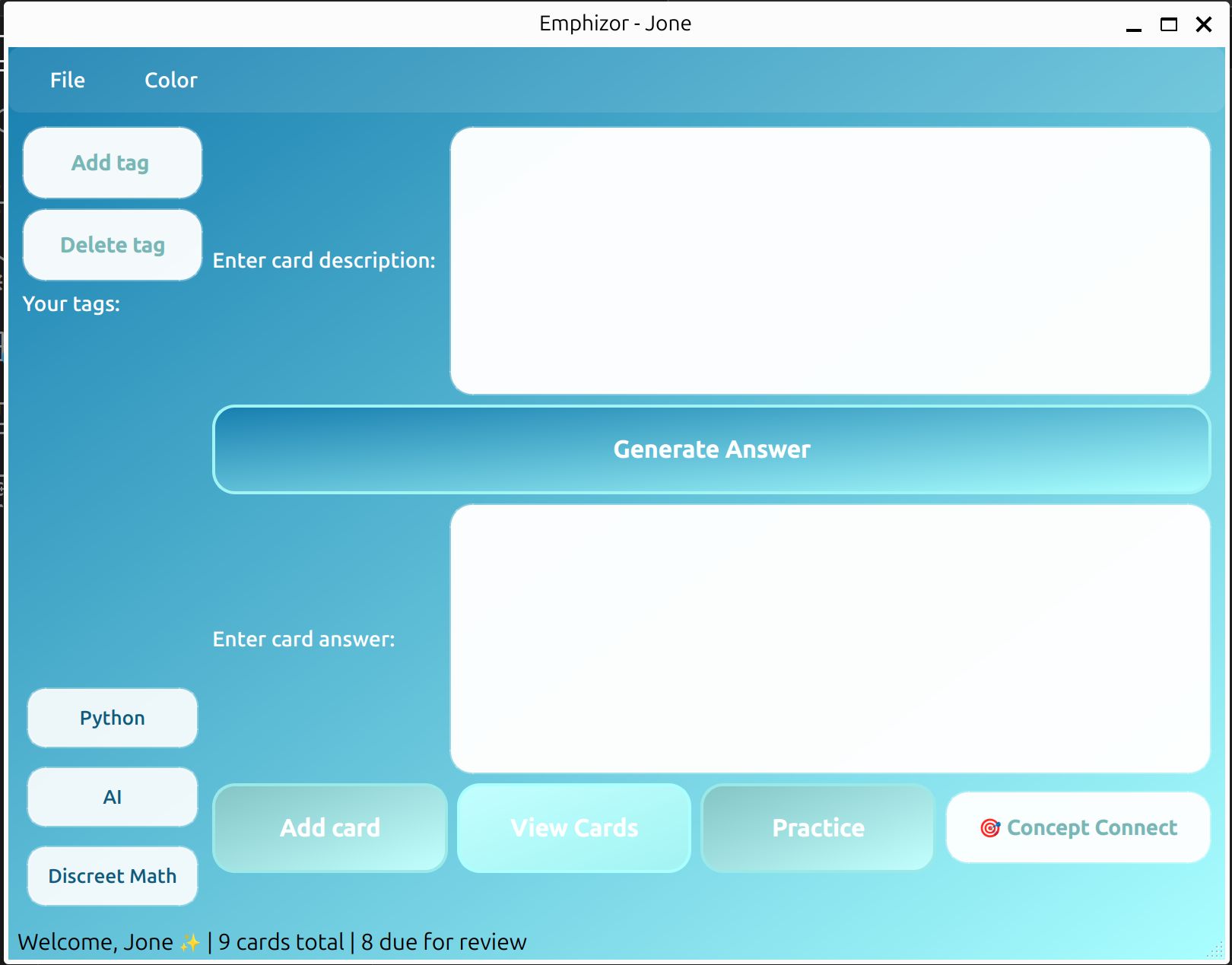
После окончания работ по этапу программирования, планируется оптимизация существующих процессов и механик для улучшения быстродействия продукта на пользовательских устройствах. Особенно большое внимание на этом этапе планируется уделить потреблению приложением оперативной памяти.

### 2.2.4 План работ по этапу «Программирование»

Для демонстрации планируемых работ по программированию в удобном виде приводится следующая таблица:

| **Даты** | **Планируемая деятельность** |
| --- | --- |
| 02.07.2025 | Программирование кнопок создания тегов, разработка механизма аутентификации пользователя |
| 04.07.2025 | Программирование кнопок создания карт, синхронизация локальных действий пользователя с облачным хранилищем |
| 05.07.2025 | Поиск и исправление багов для выхода к mvp |
| 07.07.2025 | Программирования пользовательского выбора цветовой темы приложения |
| 08.07.2025 | Программирование кнопок удаления карт и тегов |
| 09.07.2025 | Поиск багов и их исправление |
| 10.07.2025 | Оптимизация потребления оперативной памяти путём поиска и реализации более эффективных структур данных |

### 2.2.3. Дизайн Интерфйеса

По нынешней задумке, интерфейс приложения должен состоять из: поля ввода вопроса карты, поля ввода ответа карты, кнопки добавления карты, кнопки добавления тега, списка существующих тегов, кнопки генерации ответа искусственным интеллектом, кнопки для запуска режима Practice, кнопки для запуска режима Concept Connect. По предварительному черновика располагаться они должны следующим образом: 



На данном изображении изображен предварительный вариант компоновки элементов интерфейса

### 2.2.4 Разработка бэкенд части приложения

-Аутенфикация, облачное хранение информации о пользователе

### 2.2.5 Тестирование ( Testing)

Тестирование на предмет наличия багов, таких как вход в учетную запись несуществующего пользователя, отсутствие добавленных карт и т.п.

### 2.2.6 Создание Звуковых Эффектов (SFX Creation)

Создание звуковых эффектов при помощи Audacity, а также поиск уже существующих звуков в публичных сборниках.

### 2.2.7 Интеграция Звука в Приложение (Audio Integration)

Данный этап предполагает наложение созданных и найденных звуковых эффектов на действия пользователя, такие как: нажатие на кнопки, печать текста, перемещение плиток.

### 2.2.8 Интеграция и Тестирование (Integration & Testing)

Тестирование на предмет наличия багов, таких как вход в учетную запись несуществующего пользователя, отсутствие добавленных карт и т.п.

## 2.2.8.1 Альфа-Тестирование (Alpha Testing)

- Тестирование приложения внутри команды разработчиков.  
- Обнаружение и исправление ошибок.

### 2.2.8.2. Багфиксинг (Bug Fixing)

- Исправление всех обнаруженных ошибок и багов.

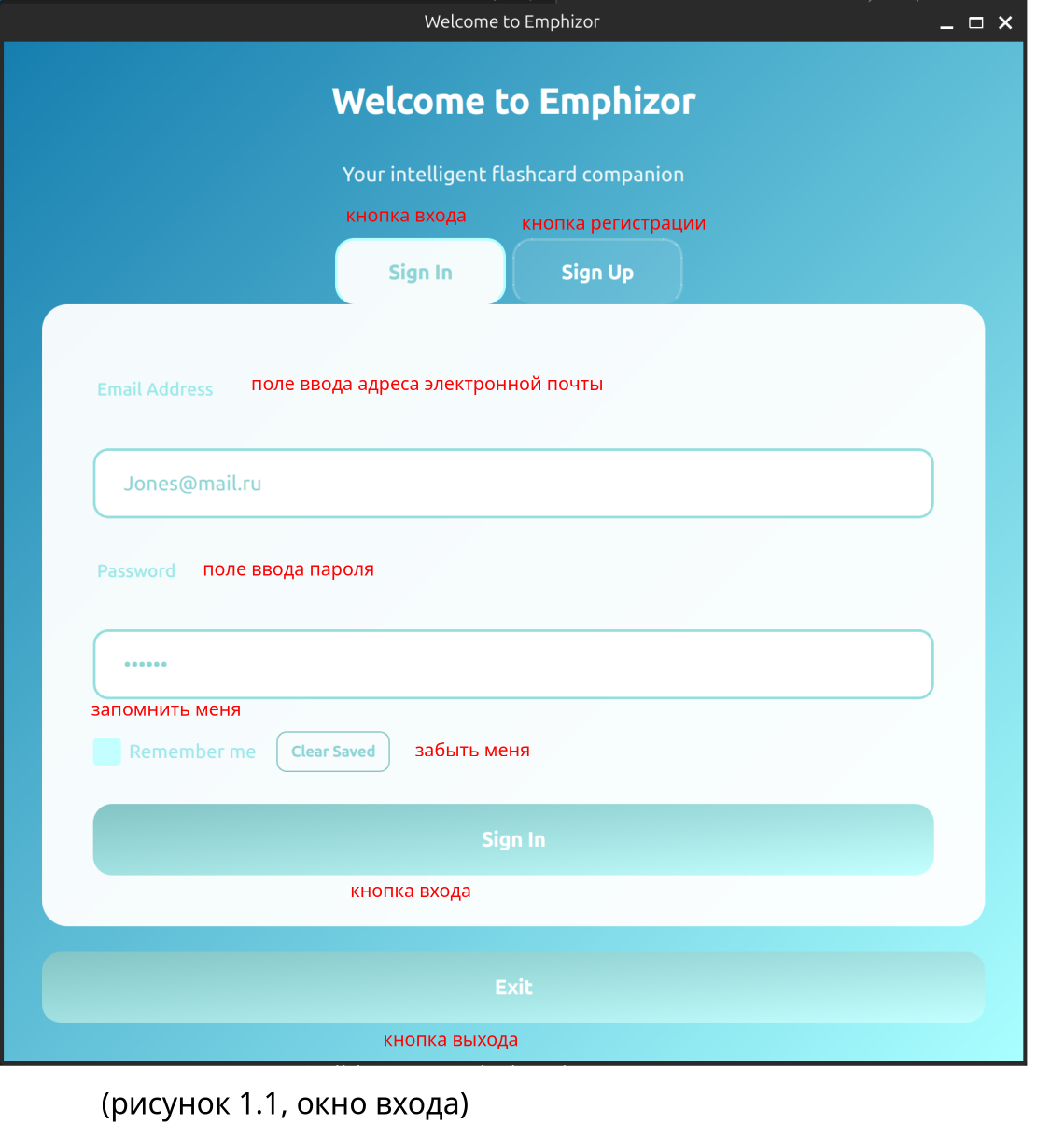
### 2.2.8.3. Оптимизация (Optimization)

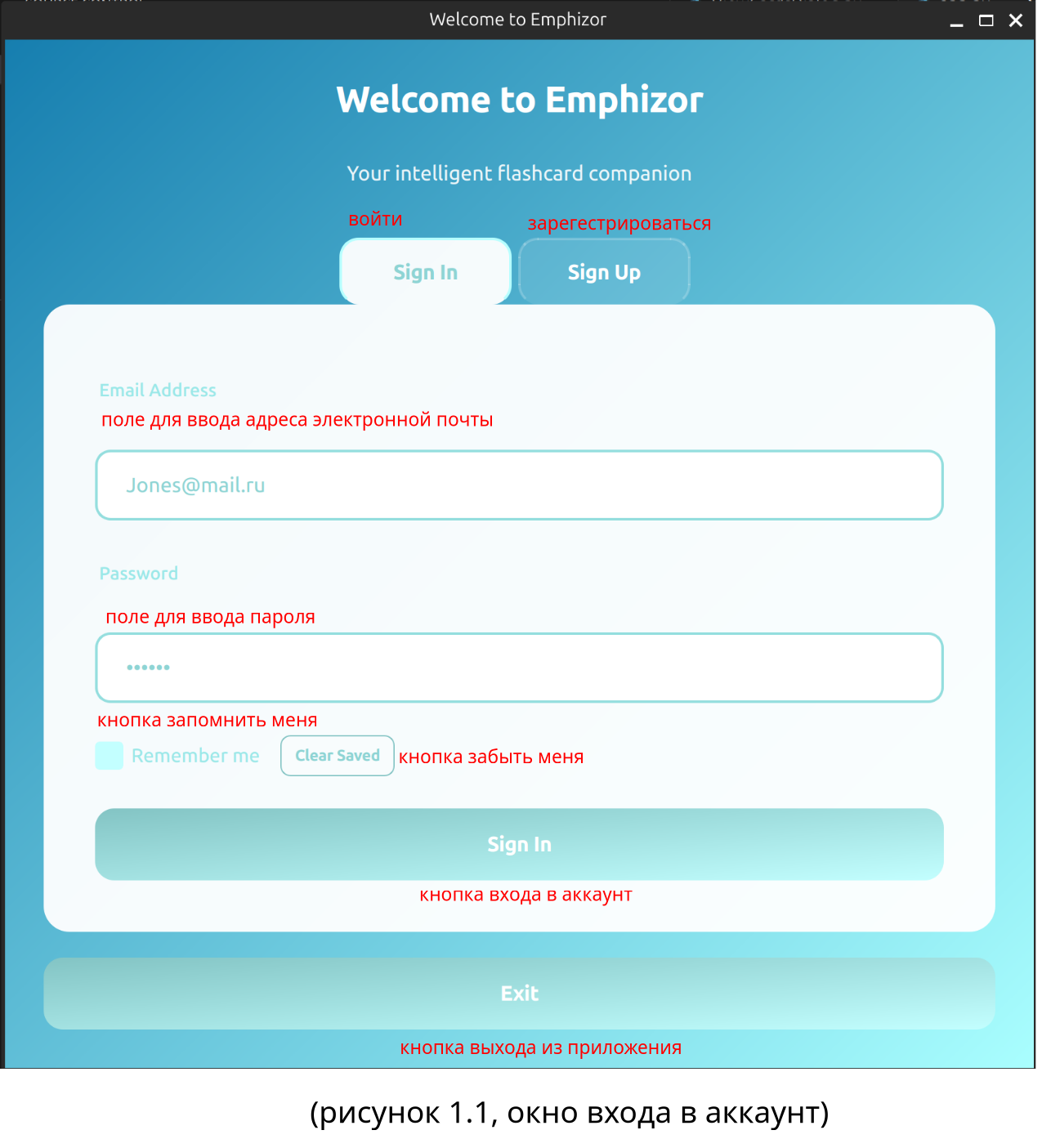
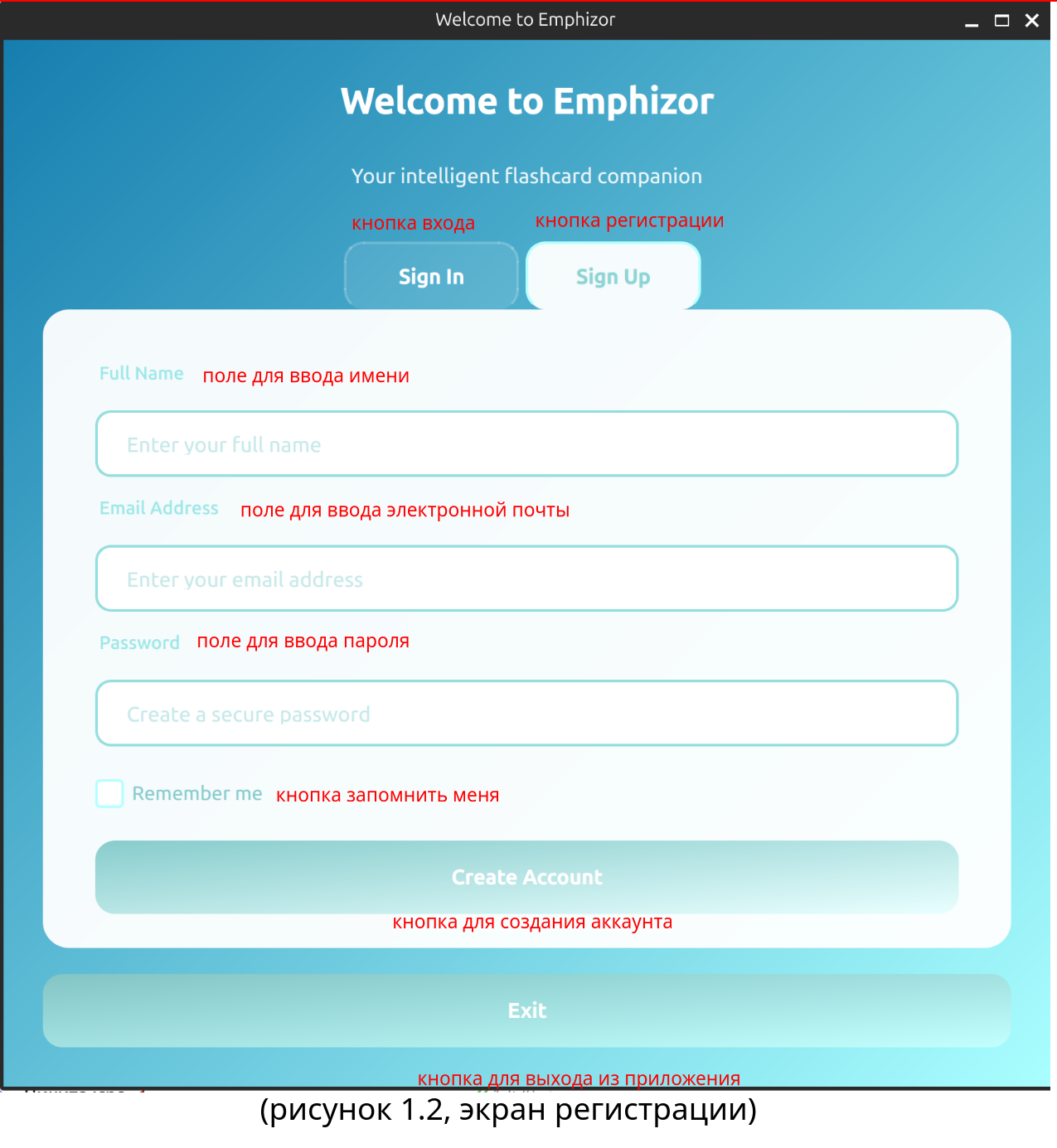
- Оптимизация производительности приложения на основе результатов тестирования.

## 

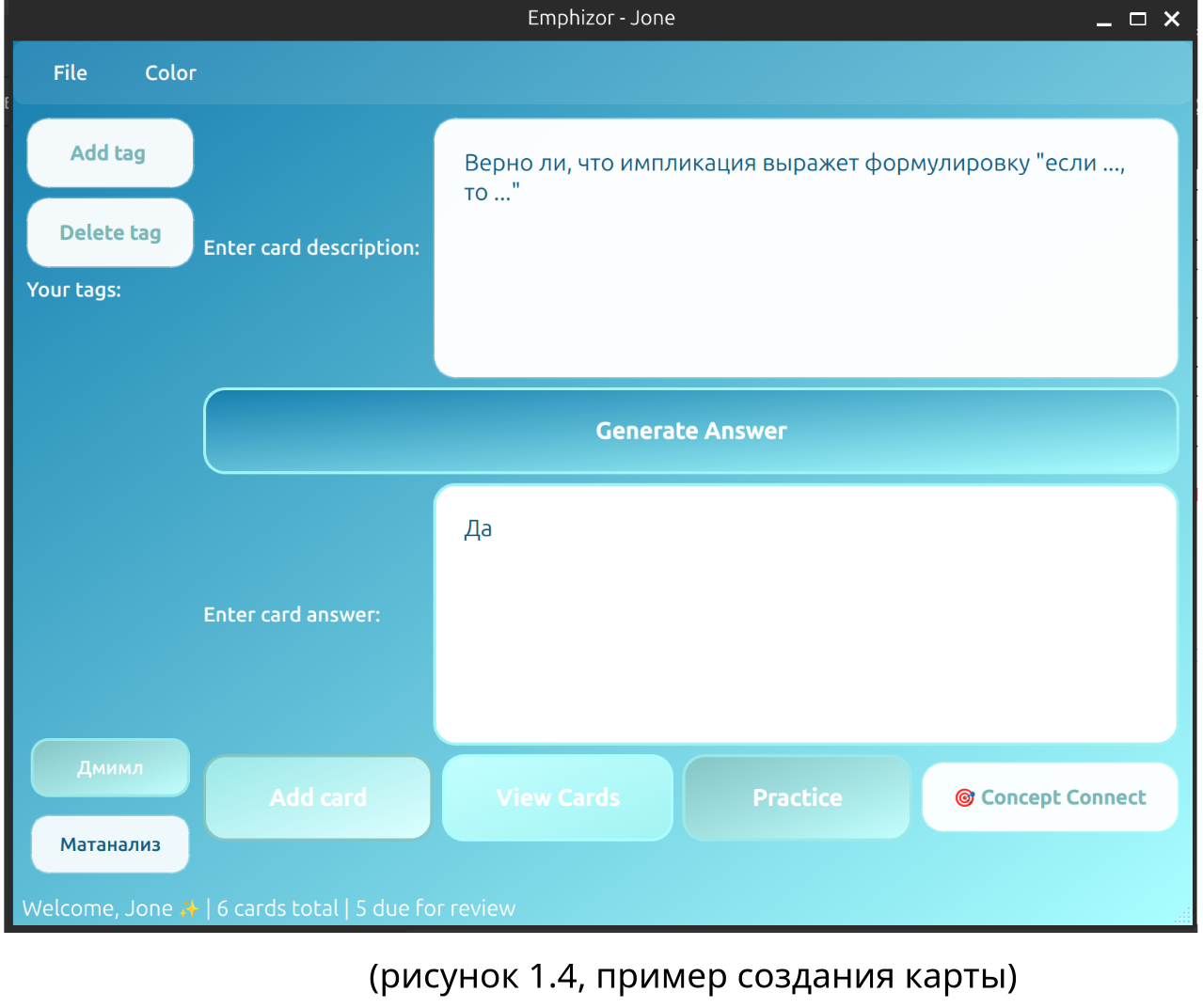
## 2.3 Интерфейс приложения

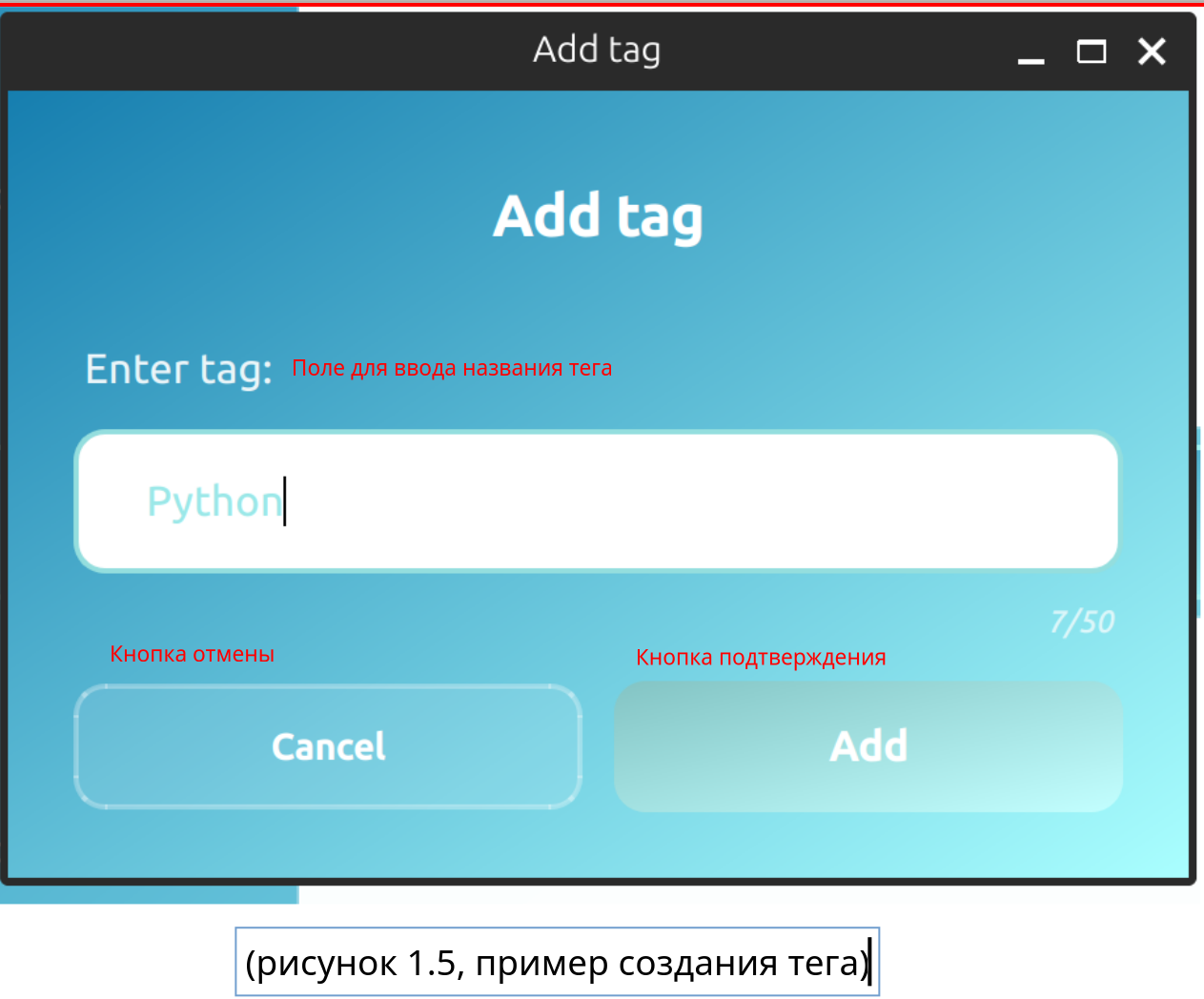
При запуске приложение показывает окно авторизации (рисунок 1.1, 1.2, 1.3).

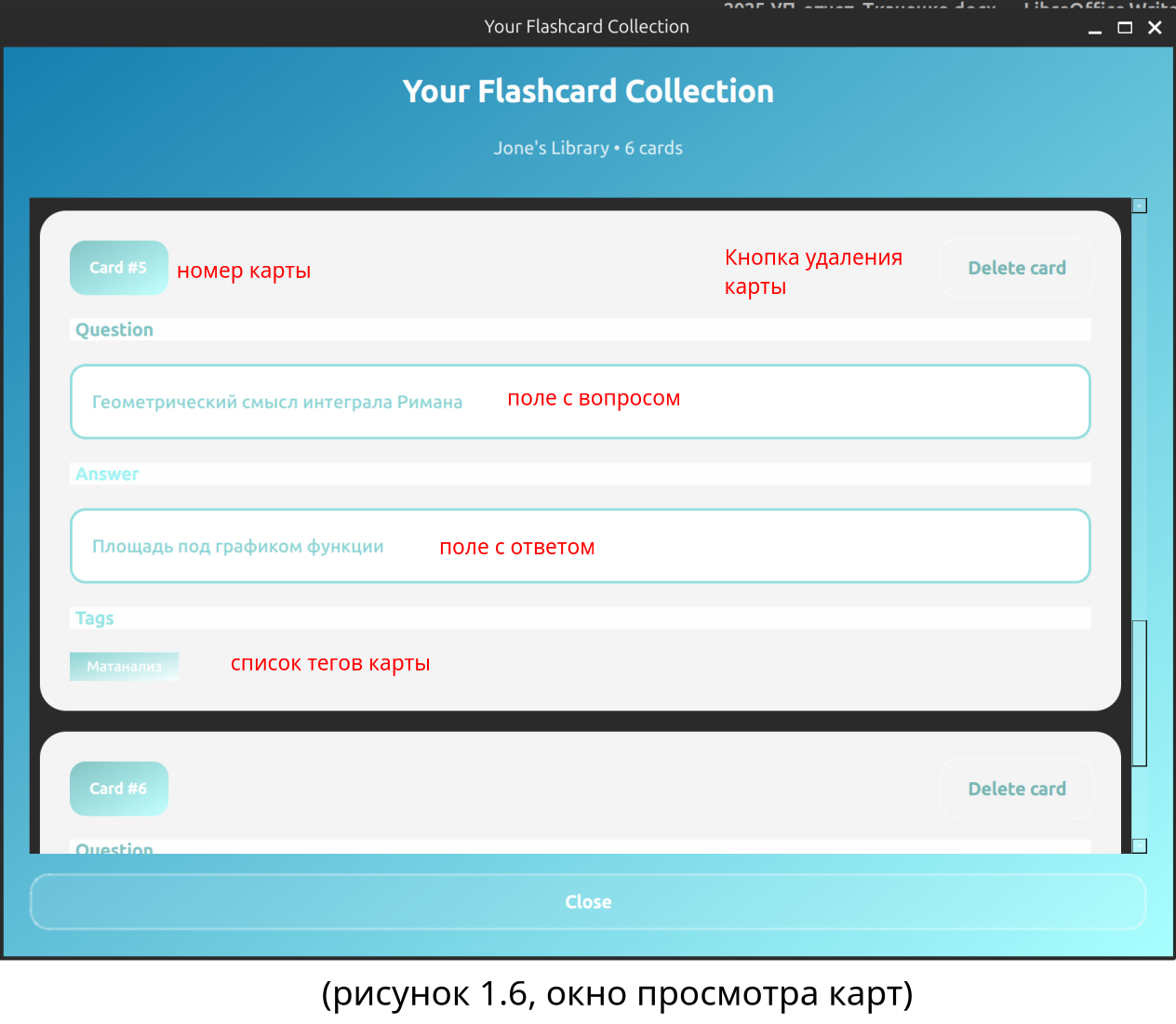


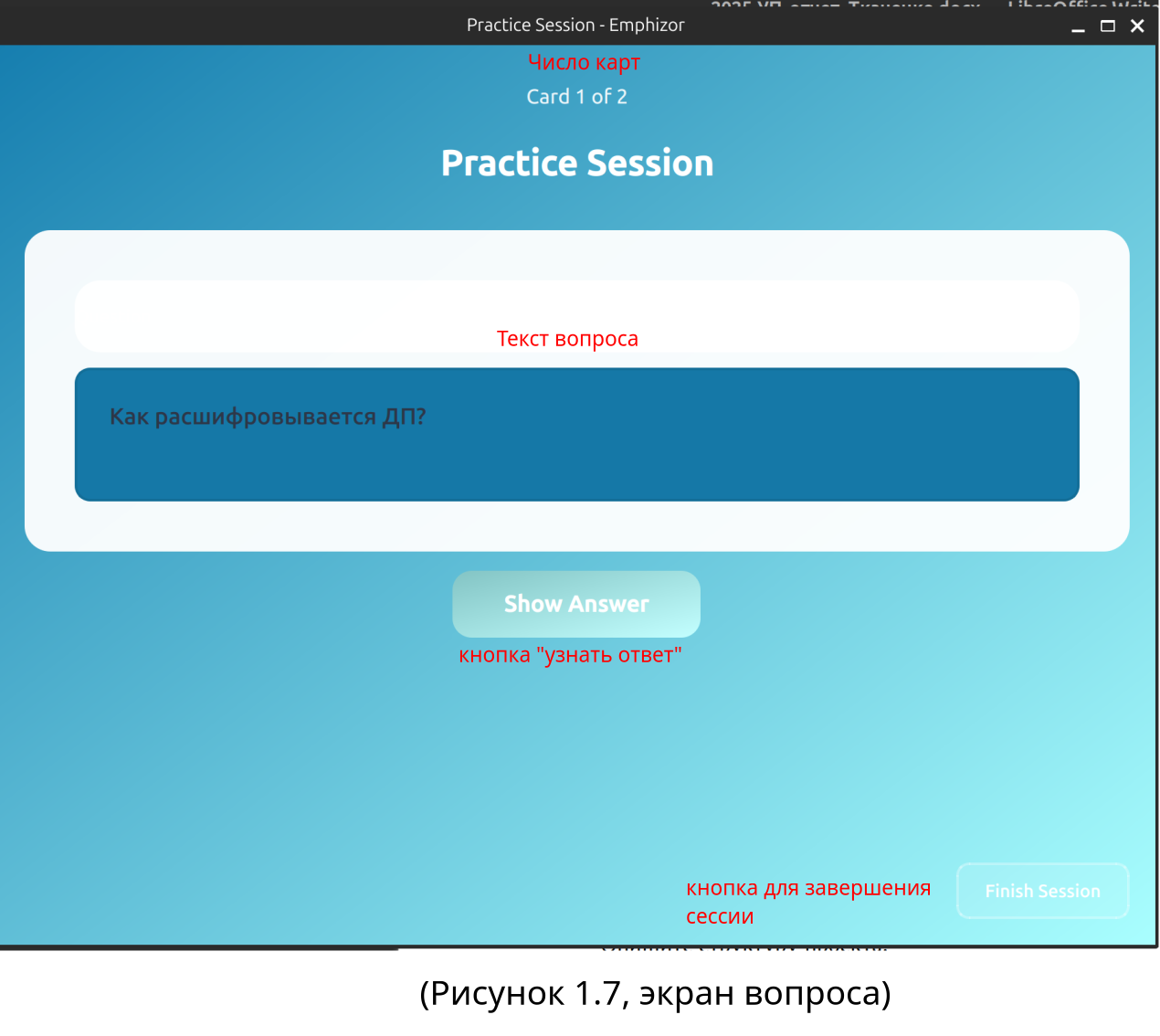


Для создания флеш карты в списке тегов, нужно выбрать те, которые относятся к карте, которую вы хотите создать. Далее необходимо вписать информацию в о наполнении карты и ответе, в соответствующие поля и нажать кнопку «Add card» (рисунок 1.4).

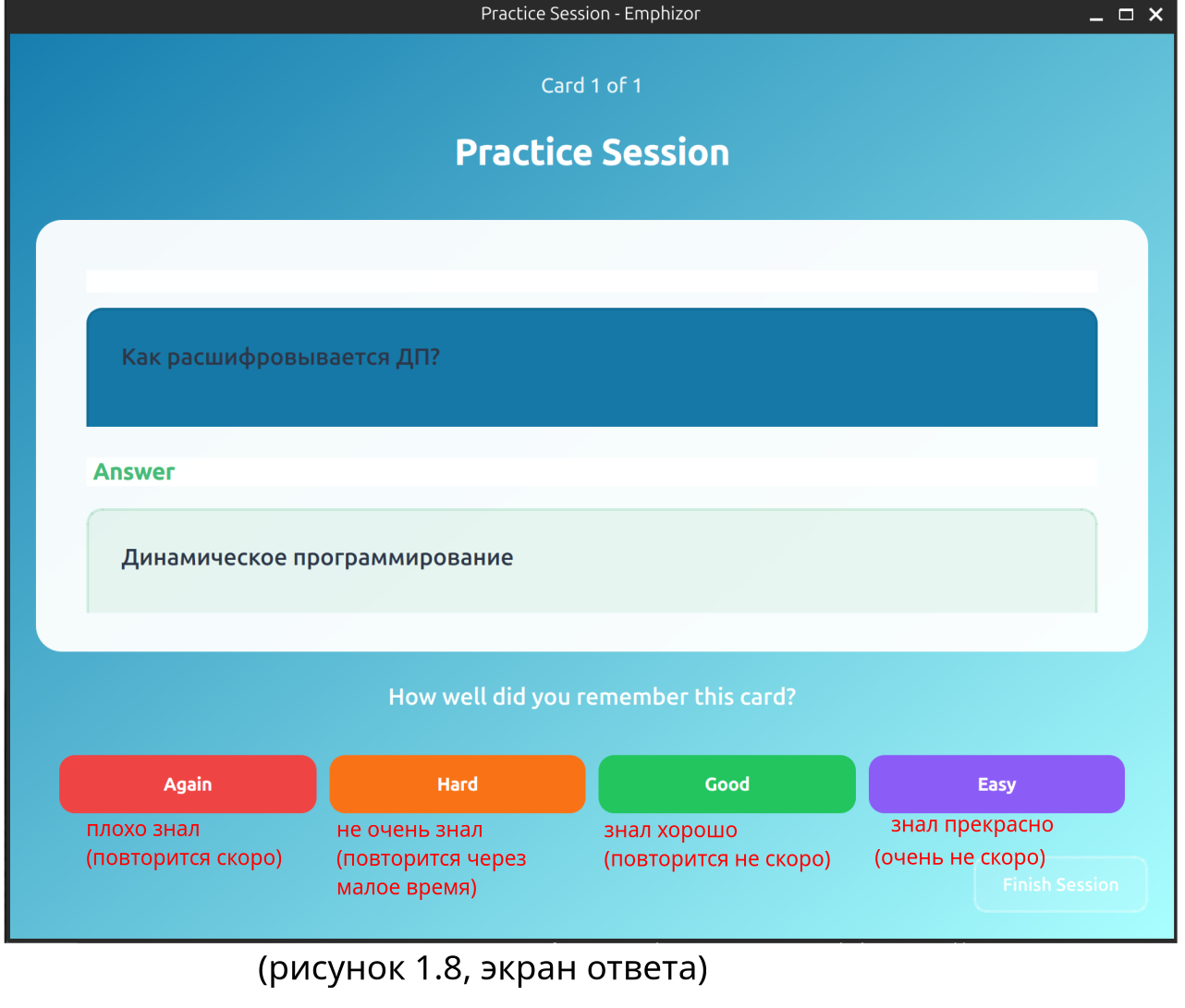
Для создания тега нажмите на кнопку «Add tag» и во всплывающем окне напишите название вашего тега и нажмите «Add» (рисунок 1.5). 

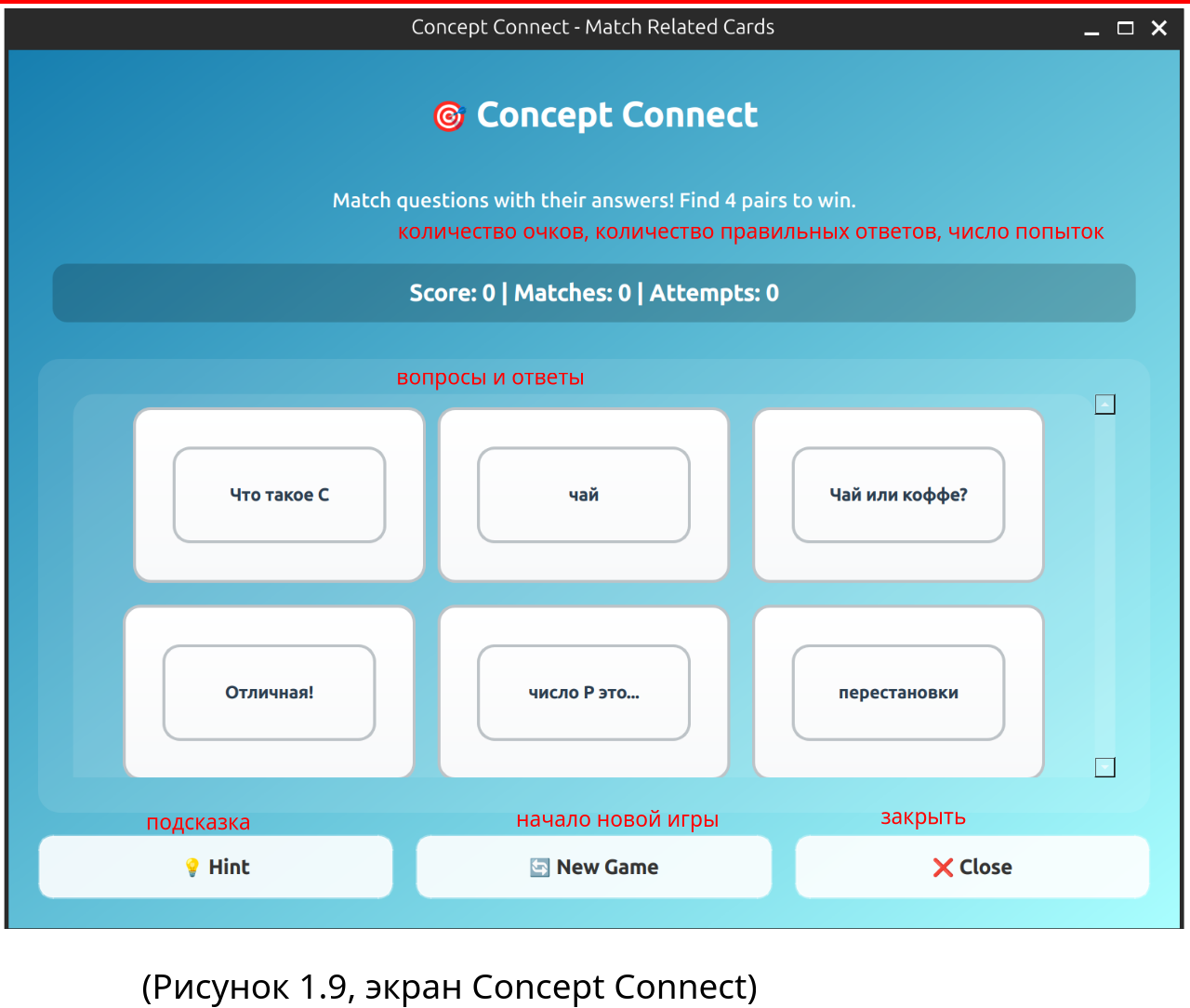
Просмотр карт: перед вами появится окно содержащее информацию о ваших картах, где вы также можете удалить лишние (рисунок 1.6).

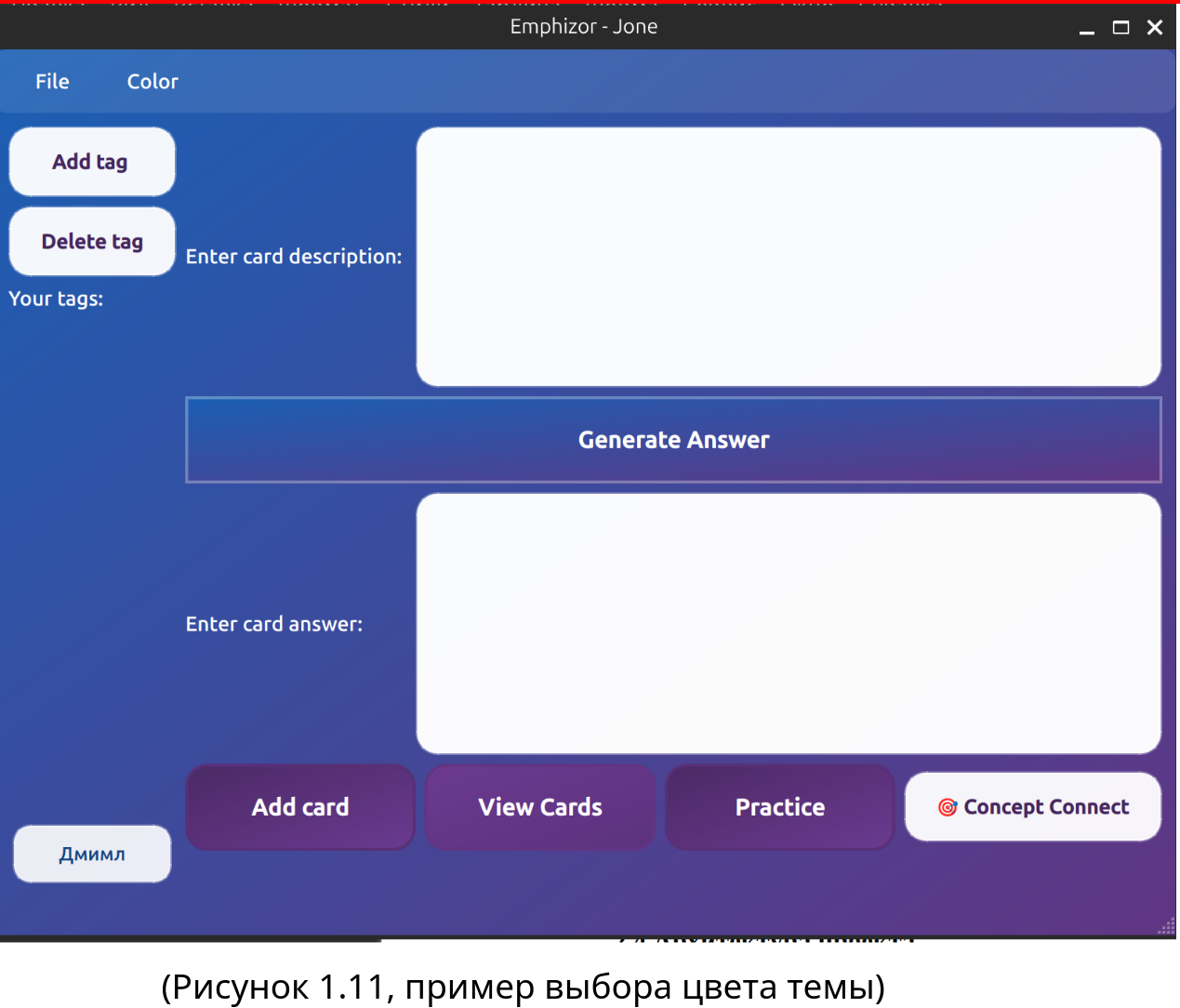
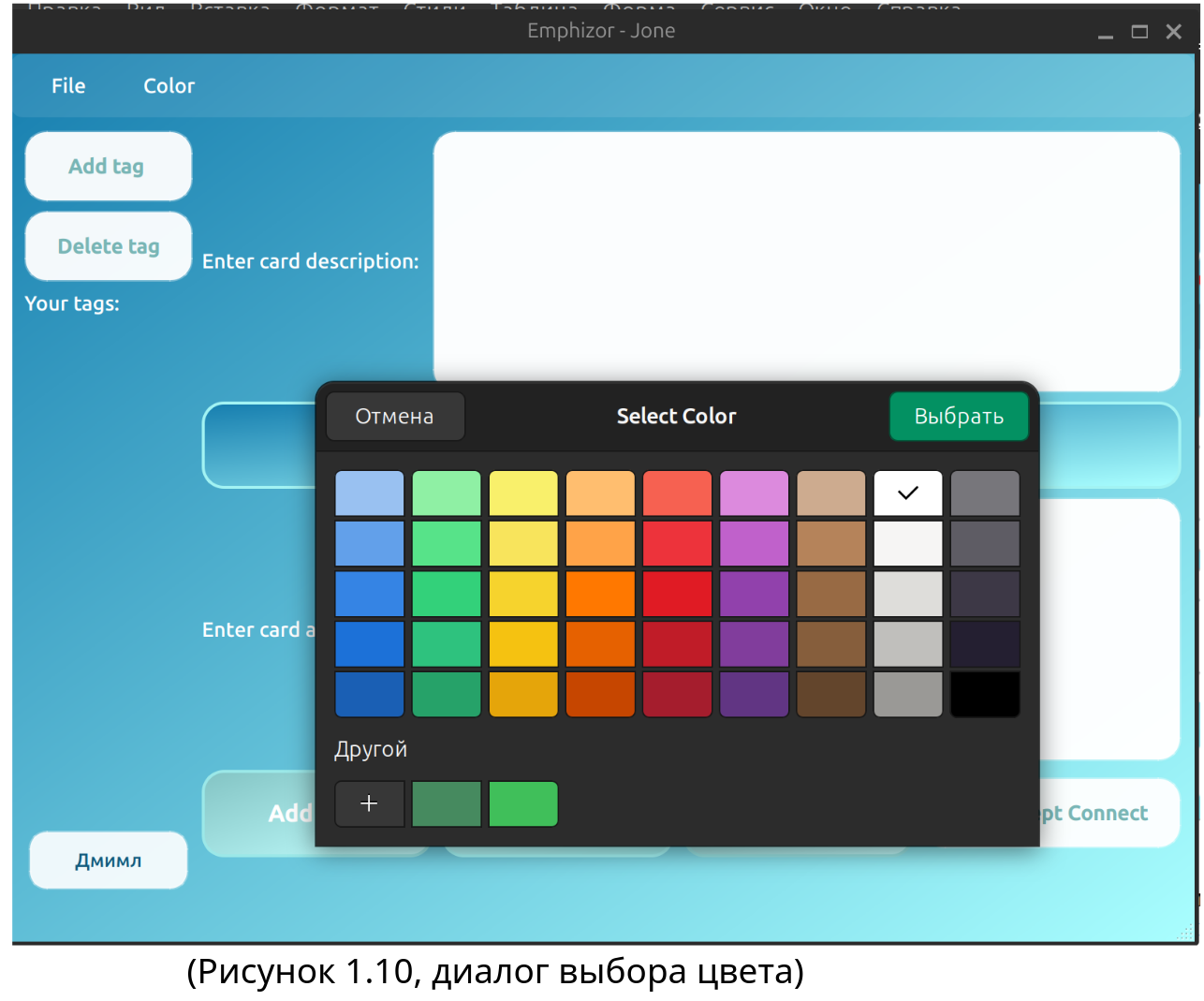
Практика созданных карт. Для этого существует два режима. Начнем с классического. Для этого нажмите на кнопки тегов, которые вы бы хотели повторить, далее нажмите на кнопку «Practice». Перед вами появится окошко, с вопросом, написанным на одной из ваших карт (рисунок 1.7).



Если нажать на «Finish session», то практика завершится. Если нажать на «Show Answer», то программа покажет вам ответ и предложит вам оценить ваше знание вопроса, от вашего ответа зависит то как часто эта карта будет далее появляться (рисунок 1.8).

Теперь попробуем второй режим тренировки, для этого нажмите в рабочей области кнопку «Concept Connect» (рисунок 1.9) 

Также приложение позволяет менять цветовую гамму, для этого в рабочей области выберите раздел «Color», затем выберите первый или второй цвет градиента вы хотите поменять. (рисунок 1.11)

Вот, например, что получится если выбрать сочетание синего и фиолетового. Таким образом вам были представлены основные детали интерфейса программы. 

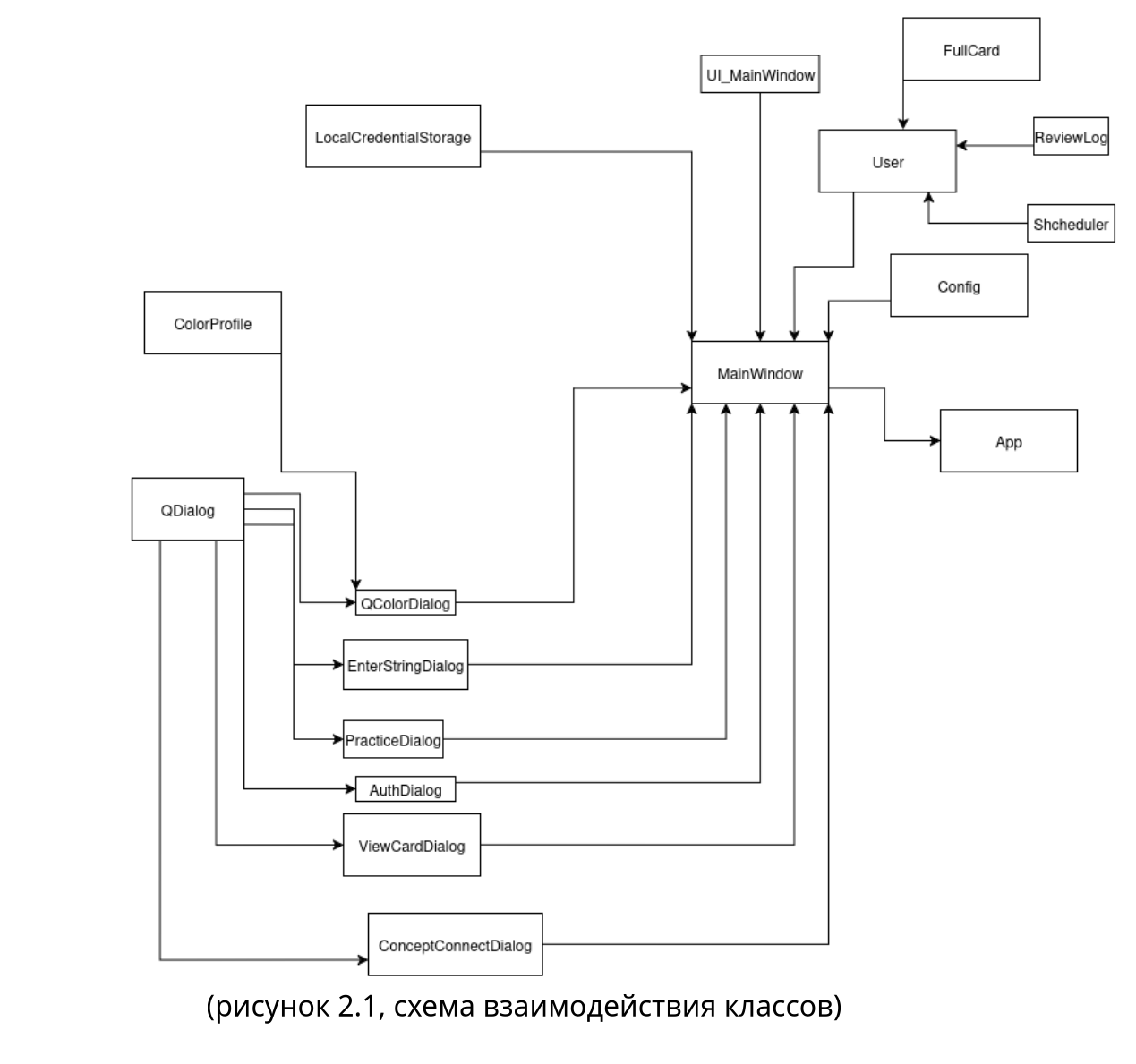
## 2.4 Архитектура проекта

В нашем проекте есть основной класс — MainWindow, на котором находятся все виджеты, кнопки и на котором располагается основная рабочая область программы.

Есть вспомогательные классы диалогов, которые отвечают за отображение и функционирование соответствующих им диалогов, например: AuthDialog, PracticeDialog, ViewCardsDialog, EnterStringDialog, ConceptConnectDialog.

Также существуют вспомогательные классы, предназначенные для хранения некоторого набора «однородных» данных, например ColorProfile, User, FullCard, Config.

Кроме них используются классы из py-fsrs, такие как: Scheduler, Card, ReviewLog. Более подробно схема их взаимодействия представлена на схеме ниже:



Стрелка от одного класса к другому обозначает «задействован в». Так например, отсюда видно, что класс ReviewLog задействован в User, а User в App (через MainWindow)

## 2.5 Реализация бизнес-логики приложения

В приложении я разрабатывал общую архитектуру а также, программную часть, отвечающую за смену цвета интерфейса, дизайн, отображение и внешний вид кнопок, отрисовку интерфейса, создание и добавление тегов, удаление тегов и карт, поиск и фикс багов и т. д.

Пример метода класса, отвечающего за удаление тега:

def delete\_tag\_button\_clicked(self):

to\_delete\_names = self.get\_selected\_tags()

for button in self.tag\_buttons[:]:

text = button.text().strip()

if text in to\_delete\_names:

idx = self.ui.verticalLayout.indexOf(button)

if idx != -1:

item = self.ui.verticalLayout.takeAt(idx)

w = item.widget()

if w:

w.setParent(None)

w.deleteLater()

self.tag\_buttons.remove(button)

for card in self.user.full\_cards:

card.tags = card.tags - to\_delete\_names

self.tags -= to\_delete\_names

Метод, отвечающий за удаление карт:

def delete\_card(self, card\_to\_delete):

self.user.full\_cards.remove(card\_to\_delete)

while self.cards\_layout.count():

item = self.cards\_layout.takeAt(0)

if item.widget():

item.widget().deleteLater()

if self.user.full\_cards:

for i, full\_card in enumerate(self.user.full\_cards):

card\_widget = self.create\_card\_widget(full\_card, i + 1)

self.cards\_layout.addWidget(card\_widget)

else:

no\_cards\_widget = QFrame()

no\_cards\_widget.setStyleSheet("""

QFrame {

background: rgba(255, 255, 255, 0.1);

border: 2px dashed rgba(255, 255, 255, 0.3);

border-radius: 20px;

padding: 30px;

margin: 15px;

}

""")

no\_cards\_layout = QVBoxLayout(no\_cards\_widget)

no\_cards\_icon = QLabel("📚")

no\_cards\_icon.setAlignment(Qt.AlignmentFlag.AlignCenter)

no\_cards\_icon.setStyleSheet("font-size: 48px; margin-bottom: 10px;")

no\_cards\_layout.addWidget(no\_cards\_icon)

no\_cards\_label = QLabel("No cards found")

no\_cards\_label.setAlignment(Qt.AlignmentFlag.AlignCenter)

no\_cards\_label.setStyleSheet("""

color: white;

font-size: 18px;

font-weight: 600;

margin-bottom: 5px;

""")

no\_cards\_layout.addWidget(no\_cards\_label)

no\_cards\_hint = QLabel("Create some flashcards to get started!")

no\_cards\_hint.setAlignment(Qt.AlignmentFlag.AlignCenter)

no\_cards\_hint.setStyleSheet("""

color: rgba(255, 255, 255, 0.7);

font-size: 14px;

font-style: italic;

""")

no\_cards\_layout.addWidget(no\_cards\_hint)

self.cards\_layout.addWidget(no\_cards\_widget)

self.cards\_layout.addStretch()

subtitle\_labels = [self.header\_layout.itemAt(i).widget() for i in range(self.header\_layout.count())]

for label in subtitle\_labels:

if isinstance(label, QLabel) and f"{self.user.name}'s Library" in label.text():

label.setText(f"{self.user.name}'s Library • {len(self.user.full\_cards)} cards")

break

## 

## 2.6 Выводы к главе 2

В ходе работы над индивидуальным проектом был получен опыт совместной разработки GUI приложения с использованием Git, Github и модель ветвления Github Flow. Работа над поддержкой регистрации и входа пользователей в приложение, хранение необходимых данных в PostgreSQL стала ценным опытом. Использование в проекте библиотек как py-fsrs, PySide6 показало на практике одну из сильных сторон Python - большое количество существующих библиотек для самых разных задач.

# Заключение

В результате индивидуальной работы было решено 150 задач по Python, изучена книга по Git, разработано приложение для интервального повторения, получен опыт работы в команде и реальный опыт использования системы контроля версий Git с использованием модели ветвления Github Flow, также получены знания в области CSS, а также изучено большое количество смежных к теме проекта тем, таких как: кривая забывания, алгоритм FSRS, PySide6.

Были получены навыки работы с UML диаграммами, редакторы для которых намного удобнее обычных графических редакторов.

В индивидуальном проекте был получен опыт распределения задач. Была разработана архитектура приложения, разработан дизайн приложения, его интерфейс, реализованы ключевые механики приложения, реализована персонализация интерфейса, была проделана работа над стилями при помощи CSS.

Были получены знания о рабочем процессе в реальных IT компаниях благодаря посещению компании «LeverX».

Были установлены цели на дальнейшее развитие проекта: добавление новых интерактивных режимов обучения, усовершенствование дизайна и пользовательского опыта, расширение возможностей пользователя для кастомизации приложения, улучшение архитектуры, работа над качеством кода, добавление в нее некоторой модульности для облегчения процесса модификации программы.

# Список использованных источников

1. <https://github.com/redbutterfly4/emphizor>
2. <https://doc.qt.io/qtforpython-6/>
3. <https://dl.acm.org/doi/10.1145/3534678.3539081?cid=99660547150>
4. <https://github.com/open-spaced-repetition/py-fsrs>
5. <https://supabase.com/>
6. <https://www.postgresql.org/>
7. <https://leverx.com/ru/>
8. <https://developers.sap.com/topics/abap-platform..html>
9. <https://git-scm.com/>
10. <https://www.epam.com/cis>
11. <https://learn.epam.com/detailsPage?id=601f195a-d408-4439-a16d-0630ed2a412e>
12. <https://education.yandex.ru/handbook/python>
13. <https://apps.ankiweb.net/>
14. <https://www.remnote.com/>