МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

Факультет *компьютерных наук*

Кафедра *программирования и информационных технологий*

***Курсовая работа***

*Мобильное приложение для изучения иностранных языков*

09.03.04 Программная инженерия

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *С.Д.Махортов, д.ф.-м.н., доцент \_\_\_. \_\_\_.20\_\_\_*

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н.А. Пеканов, 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *П.С. Лысачев, ст. преподаватель*

Воронеж 2023

**Содержание**

[Введение 4](#_heading=h.gjdgxs)

[1.](#_heading=h.30j0zll) Постановка задачи 5

[2.](#_heading=h.3znysh7) Анализ предметной области 6

[2.1 Глоссарий 6](#_heading=h.2et92p0)

[2.2 Обзор аналогов 8](#_heading=h.tyjcwt)

[2.3 Общий вывод 10](#_heading=h.3dy6vkm)

[2.4 Средства реализации 10](#_heading=h.1t3h5sf)

[2.5 Обзор Средств 11](#_heading=h.4d34og8)

[3.](#_heading=h.2s8eyo1) Реализация 11

[3.1 Архитектура и реализация классов приложения 12](#_heading=h.3j2qqm3)

[3.1.1 Системные классы запуска приложения 13](#_heading=h.1y810tw)

[3.1.2 Класс структуры данных для хранения карточек 15](#_heading=h.4i7ojhp)

[3.1.3 Классы игр 16](#_heading=h.2xcytpi)

[3.1.4 Классы пользовательского интерфейса 20](#_heading=h.1ci93xb)

[3.1.5 Классы папок и карточек для view 20](#_heading=h.3whwml4)

[3.1.6 Классы игр для view 23](#_heading=h.2bn6wsx)

[3.2 Пользовательский интерфейс 29](#_heading=h.17dp8vu)

[3.2.1. Экран папок 30](#_heading=h.3rdcrjn)

[3.2.2 Экран карточек 31](#_heading=h.26in1rg)

[3.2.3 Экран выбора игры 32](#_heading=h.lnxbz9)

[3.2.4 Экран выбора папок, которые будут участвовать в игре 33](#_heading=h.35nkun2)

[3.2.5 Экран игры (Поиск правильного перевода) 34](#_heading=h.1ksv4uv)

[3.2.6 Экран игры (Написать правильный перевод) 35](#_heading=h.2jxsxqh)

3.2.7 Экран игры (Поиск правильного перевода по аудио) 36

[3.2.8 Экран окончания игры 37](#_heading=h.z337ya)

Заключение 39

[Список использованных источников 41](#_heading=h.qsh70q)

Приложения 42

# Введение

В наше время многие люди стремятся к изучению иностранных языков. Однако, для достижения эффективных результатов необходимо иметь доступ к различным методикам и приложениям, которые помогут запоминать новые слова и фразы. Поэтому создание мобильного приложения, которое поможет пользователям изучать иностранные языки, является востребованной задачей. Основной целевой аудиторией приложения являются люди, которые изучают иностранные языки или планируют начать изучение.

Для хранения данных в приложении будет использована технология FileManager, которая обеспечит надежное хранение и организацию карточек и папок. Это позволит пользователям легко добавлять, удалять и сортировать карточки внутри папок.

Для создания пользовательского интерфейса приложения будет использован фреймворк UIKit, который предоставляет набор инструментов и компонентов для создания привлекательного и удобного интерфейса. Пользователи смогут легко перемещаться между разделами приложения, добавлять новые папки и карточки, а также использовать игры для повышения эффективности изучения языка.

В результате разработки мобильного приложения на языке Swift с использованием паттерна MVC, FileManager и UIKit, пользователи смогут эффективно изучать иностранные языки, создавать персональные папки с карточками и проверять свои знания с помощью встроенных игр.

# Постановка задачи

Цель данной курсовой работы заключается в разработке и создании полнофункционального мобильного приложения для изучения иностранных языков. Процесс разработки основан на языке программирования Swift и паттерне проектирования Model-View-Controller (MVC). Приложение будет предоставлять возможность пользователям создавать папки с карточками для изучения слов и проверять свои знания с помощью игр, связанных с изучением языков. Основная цель состоит в создании инструмента, который позволит пользователям более эффективно изучать и запоминать иностранные слова, а также улучшать навыки общения на иностранном языке.

Для успешной реализации данной цели были поставлены следующие задачи:

- Разработка функциональных возможностей мобильного приложения для изучения иностранных языков на основе языка программирования Swift и паттерна проектирования Model-View-Controller (MVC).

- Создание системы хранения и организации папок с карточками для изучения слов.

- Реализация игровых механик, позволяющих пользователям проверять и улучшать свои знания языка.

- Обеспечение удобного пользовательского интерфейса для навигации по приложению и взаимодействия с функциональными возможностями.

# Анализ предметной области

## 2.1 Глоссарий

Мобильное приложение — это специально разработанное приложение для конкретной мобильной платформы, которое обеспечивает определенный функционал или решает определенную задачу.

Верстка страницы — это процесс создания страницы, состоящей из программного кода, стилей оформления и подгружаемых картинок и фоновых изображений. Для создания верстки используется макет, который специальным образом разбивается на элементы в соответствии с дизайном.

Swift — это компилируемый язык программирования общего назначения с открытым исходным кодом, разработанный и поддерживаемый компанией Apple. Он используется для создания приложений для платформ iOS и macOS и был представлен в 2014 году.

Storyboard — это графический файл в формате XML, который используется для проектирования пользовательского интерфейса (UI) в iOS-приложениях. С помощью Storyboard разработчики могут создавать визуальные компоненты, такие как экраны, контроллеры и переходы между ними, путем перетаскивания и настройки графических элементов в интерфейсе. Storyboard позволяет разработчикам создавать более сложные UI, управлять навигацией между экранами и улучшать общую пользовательскую опытность.

UIKit — это фреймворк, разработанный компанией Apple для создания графического интерфейса пользовательского приложения для платформ iOS, iPadOS, watchOS и tvOS. Он является одним из основных инструментов для разработки приложений для платформ iOS и watchOS.

Cocoa Touch — это фреймворк, разработанный Apple для создания мобильных приложений под операционную систему iOS. Он содержит набор инструментов для создания пользовательского интерфейса, обработки ввода, работы с сетью, хранения данных и других задач, связанных с разработкой приложений для iOS.

Xcode — это интегрированная среда разработки программного обеспечения (IDE), созданная компанией Apple для разработки приложений на платформах macOS, iOS, watchOS и tvOS. Первая версия Xcode была выпущена в 2003 году. Инструментарий Xcode включает в себя средства разработки, отладки, тестирования и сборки приложений.

JSON (JavaScript Object Notation) — это легковесный формат обмена данными, используемый для передачи данных между клиентом и сервером. JSON представляет данные в виде пар "ключ-значение" и широко используется во многих языках программирования. Этот формат данных легко интерпретируется и обрабатывается, что делает его идеальным для создания веб-приложений и API.

Карточка — это элемент, который содержит информацию, используемую для запоминания и изучения определенного материала. В контексте мобильных приложений для изучения иностранных языков карточка содержит слово на одном языке и его перевод на другой язык. Карточки часто используются в методике интервальных повторений, которая помогает улучшить запоминание иностранных слов и фраз.

MVC (Model-View-Controller) — это архитектурный паттерн проектирования программного обеспечения, который широко используется в различных приложениях. MVC разделяет логику приложения на три основных компонента: модель (Model), представление (View) и контроллер (Controller). Модель отвечает за хранение данных и их обработку, представление отображает данные пользователю, а контроллер управляет взаимодействием между моделью и представлением. Это позволяет улучшить структурированность, переносимость и расширяемость кода.

## 2.2 Обзор аналогов

1. Anki

Anki – это приложение, позволяющее создавать и изучать карточки для запоминания информации, в том числе иностранных языков. В Anki можно создавать карточки с текстом, картинками, аудио и видео. Кроме того, в приложении есть возможность импортировать готовые колоды карточек.

Плюсы:

- Допускает создание карточек с разными типами контента;

- Предоставляет возможность импортировать готовые колоды карточек.

Минусы:

- Интерфейс может показаться сложным и непривычным для новых пользователей;

- Требуется время на изучение возможностей приложения.

2. Quizlet

Quizlet – это приложение для создания и изучения карточек для запоминания информации, в том числе иностранных языков. В Quizlet можно создавать карточки с текстом, картинками и аудио. Кроме того, в приложении есть возможность использовать готовые колоды карточек, созданные другими пользователями.

Плюсы:

- Доступ к готовым колодам карточек.

Минусы:

- Ограниченные возможности для создания карточек с разными типами контента;

- Отсутствие возможности добавлять видео файлы.

3. Memrise

Memrise – это приложение для изучения иностранных языков, основанное на методике запоминания слов и фраз при помощи карточек и повторений. Пользователь изучает новые слова и фразы в виде карточек, которые он повторяет в различных режимах.

Плюсы:

- Широкий выбор языков для изучения.

Минусы:

- Отсутствие возможности создания собственных карточек;

- Ограниченные возможности для настройки повторений.

4. Duolingo

Duolingo – это бесплатное приложение для изучения иностранных языков при помощи карточек и игр. В приложении доступен широкий выбор языков.

Плюсы:

- Широкий выбор языков для изучения.

Минусы:

- Карточки не всегда позволяют глубоко изучить язык;

- В некоторых языках могут содержаться ошибки перевода.

5. Rosetta Stone

Rosetta Stone - это приложение для изучения языков, которое использует интуитивный подход, исключая перевод на родной язык. В приложении представлены курсы по различным языкам, каждый из которых включает в себя уроки, задания и возможность практики произношения.

Плюсы:

• Инновационный подход к изучению языка;

• Широкий выбор языков для изучения;

• Возможность практики произношения в режиме реального времени.

Минусы:

• Достаточно высокая цена подписки;

• Отсутствие возможности создания собственных карточек и настройки курсов.

6. Lingodeer

Lingodeer — это приложение для изучения языков, которое использует геймификацию для повышения мотивации и продуктивности. В приложении представлены курсы по различным языкам, каждый из которых включает в себя уроки, задания и возможность практики произношения.

Плюсы:

• Широкий выбор языков для изучения;

• Геймификация повышает мотивацию и продуктивность.

Минусы:

• Ограниченные возможности для создания собственных карточек;

• Отсутствие возможности настройки курсов под индивидуальные потребности.

## 2.3 Общий вывод

В сравнении с представленными аналогами, приложение предлагает следующие преимущества:

1. Уникальная функция хранения папок позволяет организовать карточки пользователей более удобно и систематизировано.

2. Игры в приложении разнообразны и развивают различные навыки, включая вписывание правильного перевода и выбор правильного перевода из нескольких вариантов. Это способствует более полноценному освоению языка.

Таким образом, приложение предлагает более удобную организацию карточек и разнообразные игровые возможности, что делает его привлекательным выбором для эффективного изучения иностранных языков.

## 2.4 Средства реализации

В разработке мобильного приложения на языке Swift для изучения иностранных языков будут использованы следующие технологии:

Для клиентской части:

iOS SDK 15.0

Xcode 13.0

Swift 5.5

UIKit фреймворк для разработки пользовательского интерфейса

FileManager.default для управления хранением данных

## 2.5 Обзор Средств

Для разработки приложения мы будем использовать следующие инструменты и библиотеки:

Xcode: Xcode является основным инструментом для разработки приложений под iOS. Он обеспечивает удобную среду разработки, позволяющую создавать интерфейсы пользователя, писать код, отлаживать и тестировать приложение. Xcode также предоставляет инструменты для управления зависимостями, сборки приложения и его развертывания на устройствах.

Simulator: Симулятор, доступный в составе Xcode, позволяет разработчикам эмулировать работу приложения на различных моделях устройств iOS. Это позволяет проверить, как приложение будет выглядеть и вести себя на разных экранах и разрешениях устройств.

FileManager.default: FileManager.default предоставляет возможности для работы с файловой системой устройства. В данном случае используется FileManager.default для управления хранением данных приложения. Данные о папках и карточках, а также их сортировка, будут сохраняться в памяти устройства.

# Реализация

## 3.1 Архитектура и реализация классов приложения

Изображение выглядит как текст, диаграмма

Автоматически созданное описание

Рисунок 1 – Диаграмма прецедентов

На диаграмме прецедентов изображены основные действия пользователя в приложении:

1. Создать новую папку: Пользователь может создать новую папку для организации своих карточек. Это позволяет ему логически группировать и систематизировать свой контент.
2. Удалить папку: Пользователь имеет возможность удалить папку, если она больше не нужна. Это позволяет пользователю управлять своими папками и поддерживать актуальность информации.
3. Просмотреть список всех папок: Пользователь может просмотреть список всех доступных папок, чтобы быстро найти нужную информацию.
4. . Сортировать папки: Пользователь может сортировать папки по различным критериям, таким как алфавитный порядок или дата создания. Это помогает пользователю организовать свои папки по предпочтению.
5. Создать новую карточку: Пользователь может создать новую карточку для запоминания информации. Карточки могут содержать текст, картинки, аудио или видео.
6. Удалить карточку: Пользователь имеет возможность удалить карточку, если она больше не нужна или содержит ошибки.
7. Просмотреть список всех карточек: Пользователь может просмотреть список всех доступных карточек для быстрого доступа к необходимой информации.
8. Сортировать карточки: Пользователь может сортировать карточки по различным критериям, таким как дата создания или тематика. Это помогает пользователю организовать свои карточки для удобного изучения
9. Играть в игры: Пользователь может использовать встроенные игры для тренировки языковых навыков и запоминания информации. Это интерактивное и увлекательное средство для обучения.

Данная use-case диаграмма позволяет визуализировать взаимодействие пользователя с приложением и обозначить основные функции, которые предоставляются пользователю для эффективного использования приложения.

### 3.1.1 Системные классы запуска приложения

SceneDelegate.swift

Файл "SceneDelegate.swift" содержит реализацию класса `SceneDelegate`, который является делегатом сцены приложения. Этот класс отвечает за настройку и управление окном приложения.

Основной функционал класса SceneDelegate включает:

1. Функция scene(\_:willConnectTo:options:) вызывается при подключении сцены и используется для конфигурации и привязки окна window к сцене.

2. Функция sceneDidDisconnect(\_:) вызывается при освобождении сцены системой.

3. Функция sceneDidBecomeActive(\_:) вызывается при переходе сцены из неактивного состояния в активное состояние.

4. Функция sceneWillResignActive(\_:) вызывается при переходе сцены из активного состояния в неактивное состояние.

Класс SceneDelegate играет важную роль в управлении окном приложения и обработке событий, связанных со сценами. Он позволяет приложению настраивать окно и выполнять необходимые действия при переходе между состояниями сцен, такими как активное, неактивное и фоновое состояния.

AppDelegate.swift

Файл "AppDelegate.swift" содержит реализацию класса AppDelegate, который является делегатом приложения. Этот класс отвечает за общие настройки и управление жизненным циклом приложения.

Основной функционал класса AppDelegate включает:

1. Функция application(\_:didFinishLaunchingWithOptions:) вызывается после запуска приложения и используется для настройки и конфигурации приложения.

2. Функцияapplication(\_:configurationForConnecting:options:) вызывается при создании новой сцены и используется для выбора конфигурации сцены.

4. Функция applicationWillTerminate(\_:) вызывается перед закрытием приложения и используется для сохранения данных перед завершением приложения.

Класс AppDelegate играет важную роль в жизненном цикле приложения. Он отвечает за общие настройки и настройку приложения при запуске, а также обрабатывает события, связанные с жизненным циклом приложения, такие как создание и конфигурация сцены, сохранение данных и завершение приложения

### 3.1.2 Класс структуры данных для хранения карточек

DataStorage

Файл "DataStorage.swift" содержит реализацию структур Cell, Folder и DataStorage, которые отвечают за хранение и управление данными в приложении.

Основной функционал данного кода включает:

1. Структура Cell: Хранит информацию о слове и его переводе в свойствах word и translation.

2. Структура Folder: представляет папку, содержащую массив ячеек. Она обладает следующими возможностями:

* Хранение названия папки в свойстве name.
* Хранение массива ячеек в свойстве cells.
* Метод addCell(newCell:) для добавления новой ячейки в папку.
* Метод removeCell(at:) для удаления ячейки по индексу в папке.
* Оператор `==` для сравнения папок.

3. Структура DataStorage: представляет хранилище данных, содержащее массив папок. Её функционал включает:

- Статическое свойство shared, представляющее экземпляр DataStorage и позволяющее доступ к хранилищу данных из разных частей приложения.

* Метод folderSort() для сортировки папок по имени.
* Метод cellSort(folderIndex:) для сортировки ячеек в папке по словам.
* Метод addFolder(\_:) для добавления новой папки в хранилище.
* Метод removeFolder(at:) для удаления папки по индексу из хранилища.
* Метод getFolder(at:) для получения папки по индексу из хранилища.
* Метод updateFolder(at:with:) для обновления папки по индексу в хранилище.
* Метод saveData() для сохранения данных в файл.
* Метод loadData() для загрузки данных из файла.
* Метод dataFilePath() для получения пути к файлу с данными.
* Метод createDataFile() для создания файла данных, если он не существует.

Структуры Cell, Folder и DataStorage являются ключевыми компонентами для управления данными в приложении. Они позволяют создавать папки, добавлять и удалять ячейки, сортировать папки и ячейки, а также сохранять и загружать данные из файла. Это позволяет пользователям организовывать и управлять своими словами и переводами.

### 3.1.3 Классы игр

**Изображение выглядит как диаграмма, линия, Прямоугольник, текст

Автоматически созданное описание**

Рисунок 2 - Классы игр

**GameAbstruct**

Файл "GameAbstract.swift" содержит реализацию класса GameAbstract, который представляет абстрактную игру.

Основная функциональность данного кода заключается в следующем:

1. Класс "GameAbstract" содержит свойства allCellsForGame, maxScore и score, отвечающие за хранение информации обо всех ячейках для игры, максимальном возможном количестве очков и текущем счете соответственно.

2. Метод "getAllCells(folders:)" принимает массив папок folders в качестве параметра и возвращает массив всех ячеек из этих папок. Внутри метода используется метод flatMap для преобразования массива папок в одномерный массив ячеек.

3. Метод "shuffle()" перемешивает все ячейки в свойстве allCellsForGame с помощью метода shuffle().

4. Метод "getFirstCell()" возвращает первую ячейку из массива `allCellsForGame`.

5. Метод "scoreUp()" увеличивает значение свойства `score` на единицу.

6. Метод "gameStep(enteredWord:)" принимает введенное слово `enteredWord` в качестве параметра и возвращает булевое значение. Данный метод будет использоваться для выполнения шага в игре и проверки правильности введенного слова.

Класс `GameAbstract` представляет абстракцию игры и предоставляет базовый функционал, необходимый для работы с ячейками и управления игровым процессом. В конструкторе происходит инициализация объекта игры, включая получение всех ячеек из папок, перемешивание ячеек и установку начального значения счета. Другие методы позволяют получить первую ячейку, увеличить счет, выполнить шаг в игре и т.д. Данный класс нужен для дальнейшего наследования и реализации остальных игр.

**GameFindRigtWord**

Файл "GameFindRightWord.swift" содержит реализацию класса GameFindRightWord, который является подклассом класса GameAbstract и представляет игру по поиску правильного перевода слова.

Основная функциональность данного кода заключается в следующем:

1. Класс GameFindRightWord добавляет новое свойство indexOfSelectedCell, отвечающее за хранение индекса выбранной ячейки в игре.

2. Метод getTranslationsArray() возвращает массив переводов слов, который будет использоваться для отображения вариантов ответов в игре. Метод проверяет, что количество ячеек больше 3, затем формирует массив, включающий перевод выбранной ячейки и еще случайные переводы из остальных ячеек, не повторяющиеся. Затем массив переводов перемешивается и возвращается.

3. Метод gameStep(enteredWord:) выполняет шаг в игре. Он принимает введенное слово `enteredWord` в качестве параметра и возвращает булевое значение. Метод сравнивает введенное слово с переводом выбранной ячейки. Если слова совпадают, счет увеличивается, индекс выбранной ячейки увеличивается на 1, и метод возвращает `true`. В противном случае только индекс выбранной ячейки увеличивается на 1, и метод возвращает `false`.

Класс GameFindRightWord расширяет функциональность базового класса GameAbstract, добавляя конкретную игровую механику для игры по поиску правильного перевода слова. Он предоставляет методы для формирования вариантов ответов, выполнения шага в игре и обновления счета. Этот код может быть дальше расширен и настроен под конкретные требования и правила игры.

**GameEnterRightWord**

Файл "GameEnterRightWord.swift" содержит реализацию класса `GameEnterRightWord`, который является подклассом класса `GameAbstract` и представляет игру по вводу правильного перевода слова.

Основной функционал данного кода заключается в следующем:

1. Метод gameStep(enteredWord:) выполняет шаг в игре. Он принимает введенное слово "enteredWord" в качестве параметра и возвращает булевое значение. Метод сравнивает введенное слово с переводом первой ячейки. Если слова совпадают, счет увеличивается, первая ячейка удаляется из массива всех ячеек игры, и метод возвращает `true`. В противном случае только первая ячейка удаляется из массива, и метод возвращает `false`.

Класс GameEnterRightWord расширяет функциональность базового класса GameAbstract, добавляя конкретную игровую механику для игры по вводу правильного перевода слова. Он предоставляет метод для выполнения шага в игре, обновления счета и удаления использованной ячейки.

**GameAudioTranslation**

Файл "GameAudioTranslation.swift" содержит реализацию класса GameAudioTranslation, который является подклассом класса GameAbstract и представляет игру по аудиопереводу.

Основнаяфункциональность данного кода заключается в следующем:

1. Метод getTranslationsArray()возвращает массив переводов, который будет использоваться в игре. Метод получает все ячейки игры и выбирает перевод из первой выбранной ячейки. Затем метод случайным образом выбирает еще три перевода из остальных ячеек и перемешивает массив переводов. Если количество ячеек в игре меньше 4, метод возвращает `nil`, так как для игры требуется как минимум 4 перевода.

2. Метод gameStep(enteredWord:) выполняет шаг в игре. Он принимает введенное слово enteredWord в качестве параметра и возвращает булевое значение. Метод сравнивает введенное слово с переводом первой выбранной ячейки. Если слова совпадают, счет увеличивается, индекс выбранной ячейки увеличивается на 1, и метод возвращает `true`. В противном случае только индекс выбранной ячейки увеличивается на 1, и метод возвращает `false`.

Класс GameAudioTranslation расширяет функциональность базового класса GameAbstract, добавляя конкретную игровую механику для игры по аудиопереводу. Он предоставляет методы для получения массива переводов, выполнения шага в игре, обновления счета и увеличения индекса выбранной ячейки.

### 3.1.4 Классы пользовательского интерфейса

**Изображение выглядит как диаграмма, План, снимок экрана, линия

Автоматически созданное описание**

Рисунок 3- Классы пользовательского интерфейса

### 3.1.5 Классы папок и карточек для view

**MainViewController:**

Файл "MainViewController.swift" содержит реализацию класса MainViewController, который является подклассом UIViewController и обрабатывает основную логику главного экрана приложения.

Основной функционал кода:

1. MainViewController реализует протоколы UITableViewDelegate и UITableViewDataSource, чтобы обрабатывать события и предоставлять данные для таблицы folderTableView.

2. Метод viewDidLoad() настраивает таблицу folderTableView с помощью метода setupTableView().

3. Метод setupTableView() устанавливает делегат и источник данных для таблицы folderTableView, позволяя MainViewController обрабатывать события и предоставлять данные для отображения.

4. Метод sortFolders(\_:) вызывается при нажатии на кнопку "Sort" и сортирует папки в DataStorage.shared, а затем обновляет таблицу folderTableView.

5. Метод addNewFolder(\_:) вызывается при нажатии на кнопку "Add" и отображает диалоговое окно, в котором пользователь может ввести название новой папки. После нажатия кнопки "Add" создается новый объект Folder с указанным названием и пустым массивом ячеек. Этот объект добавляется в DataStorage.shared, а затем таблица folderTableView обновляется.

6. Реализация протокола UITableViewDataSource включает методы, определяющие количество секций и количество строк в таблице, а также создающие и настраивающие ячейки для каждой строки на основе данных из DataStorage.shared.

7. Реализация протокола UITableViewDelegate включает метод tableView(\_:didSelectRowAt:), который вызывается при выборе строки в таблице. В данном случае он переходит к другому представлению (экрану) с идентификатором "showCards", передавая выбранный индекс папки.

8. Метод prepare(for:sender:) вызывается перед переходом к другому представлению (экрану) и передает данные (в данном случае, индекс выбранной папки) в целевой контроллер назначения (CardTableViewController).

9. Метод tableView(\_:commit:forRowAt:) вызывается при удалении строки из таблицы путем смахивания влево. Он удаляет соответствующую папку из DataStorage.shared и обновляет таблицу folderTableView.

Класс MainViewController является контроллером главного экрана приложения и отвечает за отображение списка папок и обработку событий, таких как сортировка, добавление и удаление папок.

Этот код может быть дальше расширен и настроен под конкретные требования и дизайн приложения.

**CardTableViewController:**

Файл "CardTableViewController.swift" содержит реализацию класса `CardTableViewController`, который является подклассом `UITableViewController` и отвечает за отображение и управление списком ячеек (карточек) внутри выбранной папки.

Основной функционал кода:

1. CardTableViewController содержит свойство folderIndex, указывающее на индекс выбранной папки в хранилище данных DataStorage.shared.

2. Метод viewDidLoad() выполняет необходимые инициализации при загрузке экрана.

3. Метод sortCell(\_:) вызывается при нажатии на кнопку "Sort" и сортирует ячейки в выбранной папке folderIndex в хранилище данных DataStorage.shared. Затем метод обновляет таблицу tableView, чтобы отразить изменения.

4. Метод addNewCell(\_:) вызывается при нажатии на кнопку "Add" и отображает диалоговое окно, где пользователь может ввести новое слово и его перевод. После нажатия кнопки "Add" создается новый объект `Cell` с указанными словом и переводом. Этот объект добавляется в выбранную папку folderIndex в хранилище данных DataStorage.shared, а затем таблица tableView обновляется.

5. Реализация протокола UITableViewDataSource включает методы, определяющие количество секций и количество строк в таблице, а также создающие и настраивающие ячейки для каждой строки на основе данных из выбранной папки в хранилище данных DataStorage.shared.

6. Реализация протокола UITableViewDelegate включает метод tableView(\_:commit:forRowAt:), который вызывается при удалении строки из таблицы путем смахивания влево. Он удаляет соответствующую ячейку из выбранной папки в хранилище данных DataStorage.shared и обновляет таблицу tableView.

Класс CardTableViewController является контроллером экрана, отображающего список ячеек (карточек) внутри выбранной папки. Он отвечает за отображение данных, обработку событий пользователей (например, добавление, сортировка и удаление ячеек) и взаимодействие с хранилищем данных.

### 3.1.6 Классы игр для view

**AllGamesViewController**

Файл "AllGamesViewController.swift" содержит реализацию класса AllGamesViewController, который является подклассом UIViewController и отвечает за отображение списка доступных игр и переход к настройкам выбранной игры.

Основной функционал данного кода заключается в следующем:

1. Класс AllGamesViewController содержит свойство gameName, которое хранит имя выбранной игры.

2. Метод viewDidLoad() вызывается при загрузке экрана и выполняет дополнительные настройки представления, если необходимо.

3. Метод findRightWordButton(\_:), enterRightWordButton(\_:) и audioTranslationButton(\_:) вызываются при нажатии на соответствующие кнопки игр ("FindRightWord", "EnterRightWord" и "AudioTranslation") и устанавливают соответствующее имя игры в свойство `gameName`. Затем они выполняют переход к экрану настроек игры, используя метод performSegue(withIdentifier:sender:).

4. Метод prepare(for:sender:) вызывается перед переходом на экран настроек игры. Он проверяет, что идентификатор перехода соответствует "GoToGameSettings" и что отправитель является экземпляром UIButton. Если условия выполняются, метод получает экземпляр GameSettingsViewController в качестве целевого контроллера перехода и передает ему имя выбранной игры в свойство gameName.

Класс AllGamesViewController отвечает за отображение списка доступных игр и обработку нажатий кнопок для выбора игры. Он также осуществляет передачу информации о выбранной игре на экран настроек игры.

**GameSettingsViewController**

Файл "GameSettingsViewController.swift" содержит реализацию класса GameSettingsViewController, который является подклассом UIViewController и отвечает за настройку параметров игры и переход к выбранной игре.

Основной функционал данного кода заключается в следующем:

1. Класс GameSettingsViewController содержит свойства selectedFolders и gameName, которые хранят выбранные папки и имя выбранной игры соответственно.

2. Метод viewDidLoad() вызывается при загрузке экрана и выполняет инициализацию и настройку элементов интерфейса. Он также устанавливает кнопку "Go To Game" в неактивное состояние.

3. Метод isFoldersSelected() проверяет, выбраны ли папки для игры. Он проверяет, что массив `selectedFolders` не пустой и содержит более 4 ячеек (по условию). Если условия выполняются, метод возвращает `true`, в противном случае - `false`.

4. Метод goToGameButtonPressed(\_:) вызывается при нажатии на кнопку "Go To Game". Он осуществляет переход к выбранной игре, исходя из значения свойства gameName, используя метод performSegue(withIdentifier:sender:).

5. Метод prepare (for:sender:) вызывается перед переходом на экран игры. Он проверяет идентификатор перехода и инициализирует соответствующий контроллер игры (GameFindRihtWordViewController, GameEnterRightWordViewController или GameAudioTranslationViewController). Затем метод передает выбранные папки в экземпляр игры.

6. Метод switchButtonsEnabled() включает или выключает кнопку "Go To Game" в зависимости от выбранных папок. Если выбраны подходящие папки (соответствующие условиям), кнопка становится активной, в противном случае - неактивной.

7. Реализация протоколов UITableViewDataSource и UITableViewDelegate обеспечивает отображение списка папок и их выбор в таблице. Методы tableView(\_:numberOfRowsInSection:) и tableView(\_:cellForRowAt:)` определяют количество ячеек и настройку каждой ячейки, соответственно. Метод tableView(\_:didSelectRowAt:) вызывается при выборе папки и осуществляет добавление или удаление выбранной папки из массива selectedFolders. После этого происходит обновление таблицы и проверка активности кнопки "Go To Game".

Класс "GameSettingsViewController" отвечает за настройку параметров игры и выбор папок для игры. Он также осуществляет передачу информации о выбранных папках.

**GameFindRigtWordViewController**

Файл "GameFindRihtWordViewController.swift" содержит реализацию класса "GameFindRihtWordViewController", который является подклассом "UIViewController" и отвечает за отображение игры "FindRight Word".

Основной функционал данного кода заключается в следующем:

1. Класс "GameFindRihtWordViewController" содержит свойства game и buttons. game представляет экземпляр игры "FindRight Word", а buttons представляет массив кнопок, используемых для выбора слов.

2. Метод "viewDidLoad()" вызывается при загрузке экрана и выполняет инициализацию и настройку элементов интерфейса. Он добавляет обработчики нажатия на кнопки и вызывает метод "formUpdate()".

3. Метод "buttonTapped(\_:)" вызывается при нажатии на кнопку. Он отключает все кнопки, проверяет правильность выбранного слова и обновляет соответствующие элементы интерфейса. Затем метод запускает таймер, который после задержки обновляет экран игры или переходит к результатам, в зависимости от текущего состояния игры.

4. Метод "prepare(for:sender:)" вызывается перед переходом на экран результатов игры. Он передает экземпляр игры "self.game" в контроллер результатов.

5. Метод "formUpdate()" обновляет состояние экрана игры. Он проверяет, что игра и массив ячеек для игры не пустые и содержат более 4 элементов. Затем метод обновляет заголовки кнопок с переводами слов, отображает текущий счет и слово для перевода, а также скрывает метку с результатом проверки.

Класс "GameFindRihtWordViewController" отвечает за отображение игры "FindRight Word". Он обрабатывает нажатия на кнопки выбора слов, обновляет состояние игры и переходит к результатам после завершения игры.

**GameEnterRightWordViewController**

Файл "GameEnterRightWordViewController.swift" содержит реализацию класса "GameEnterRightWordViewController", который является подклассом "UIViewController" и отвечает за отображение игры "Enter Right Word".

Основной функционал данного кода заключается в следующем:

1. Класс "GameEnterRightWordViewController" содержит свойство game, которое представляет экземпляр игры "Enter Right Word".

2. Метод "viewDidLoad()" вызывается при загрузке экрана и выполняет инициализацию и настройку элементов интерфейса. Он устанавливает начальное значение счета, отображает первое слово для перевода и скрывает метку с результатом проверки.

3. Метод "CheckTranslation(\_:)" вызывается при нажатии на кнопку проверки перевода. Он получает введенный перевод из текстового поля, проверяет его на правильность и обновляет соответствующие элементы интерфейса. Затем метод запускает таймер, который после задержки обновляет экран игры или переходит к результатам, в зависимости от текущего состояния игры.

4. Метод "prepare(for:sender:)" вызывается перед переходом на экран результатов игры. Он передает экземпляр игры "self.game" в контроллер результатов.

Класс"GameEnterRightWordViewController"отвечает за отображение игры "EnterRightWord". Он обрабатывает введенные переводы, обновляет состояние игры и переходит к результатам после завершения игры.

**GameAudioTranslationViewController**

Описание кода для курсовой работы:

Файл "GameAudioTranslationViewController.swift" содержит реализацию класса "GameAudioTranslationViewController", который является подклассом "UIViewController" и отвечает за отображение игры "Audio Translation".

Основной функционал данного кода заключается в следующем:

1. Класс "GameAudioTranslationViewController" содержит свойство game, которое представляет экземпляр игры "Audio Translation".

2. Метод "viewDidLoad()" вызывается при загрузке экрана и выполняет инициализацию и настройку элементов интерфейса. Он устанавливает обработчики для кнопок, которые вызывают метод "buttonTapped(\_:)" при нажатии, и обновляет интерфейс игры.

3. Метод "PlayTranslation(\_:)" вызывается при нажатии на кнопку воспроизведения звука перевода. Он использует "AVSpeechSynthesizer" для воспроизведения звука текущего слова игры.

4. Метод "buttonTapped(\_:)" вызывается при нажатии на одну из кнопок с вариантами перевода. Он обрабатывает выбор игрока, обновляет состояние игры и переходит к следующему слову или результатам игры после задержки.

5. Метод "prepare(for:sender:)" вызывается перед переходом на экран результатов игры. Он передает экземпляр игры "self.game" в контроллер результатов.

6. Метод "formUpdate()" обновляет интерфейс игры. Он устанавливает тексты кнопок в соответствии с вариантами перевода, обновляет счет, устанавливает текущее слово для перевода и скрывает метку с результатом проверки.

Класс"GameAudioTranslationViewController"отвечаетзаотображениеигры "Audio Translation". Он позволяет игроку выбирать варианты перевода, воспроизводить звуковое представление слова и обновлять состояние игры.

**ScoreViewController**

Описание кода для курсовой работы:

Файл "ScoreViewController.swift" содержит реализацию класса `ScoreViewController`, который является подклассом `UIViewController` и отвечает за отображение результатов игры.

Основной функционал данного кода заключается в следующем:

1. Класс "ScoreViewController" содержит свойство game, которое представляет экземпляр абстрактной игры (GameAbstract), для которой отображаются результаты.

2. Метод "viewDidLoad()" вызывается при загрузке экрана и выполняет инициализацию и настройку элементов интерфейса. Он скрывает кнопку "Назад" в навигационной панели и отображает текущий счет игрока и максимальный возможный счет.

3. Метод "GoToAllGames(\_:)" вызывается при нажатии на кнопку "Все игры". Он выполняет переход к экрану, отображающему все доступные игры, путем возвращения к предыдущему контроллеру в стеке навигации.

Класс "ScoreViewController" отвечает за отображение результатов игры и позволяет игроку вернуться к списку всех игр.

## 3.2 Пользовательский интерфейс

Изображение выглядит как диаграмма, текст, План, Технический чертеж

Автоматически созданное описание

Рисунок 4 – Диаграмма экранов приложения

Диаграмма экранов приложения представляет собой визуальное представление структуры и навигации внутри приложения. Она показывает связи между различными экранами, их последовательность и взаимодействие с пользователем.

### 3.2.1. Экран папок

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 5 - Экран папок

На странице "Все папки" располагается список всех ранее созданных папок. Для удобства пользователей имеются кнопки сортировки папок в алфавитном порядке и добавления новой папки.

Для создания новой папки необходимо нажать на кнопку в верхней панели, которая представляет собой символ "+" — это вызовет выпадающее окно с полем для ввода имени папки, а также кнопками "Добавить" и "Отменить". Чтобы удалить папку, нужно смахнуть ее влево.

Таким образом, на странице "Все папки" пользователи могут легко управлять своими папками, добавлять новые, сортировать и удалять их, что обеспечивает более удобный и эффективный опыт использования.

### 3.2.2 Экран карточек

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 6 - Экран карточек

Страница со списком карточек выбранной папки содержит перечень всех ранее созданных карточек. Если карточек еще не было добавлено в папку, список будет пустым. В верхней части страницы расположены кнопки, позволяющие сортировать список карточек в алфавитном порядке, а также добавлять новые карточки.

Для добавления новой карточки на странице находится кнопка в верхней панели, представленная символом «плюс». При ее нажатии появляется выпадающее окно с полями для ввода слова и его перевода, а также кнопками «Добавить» и «Отмена».

Удаление карточек осуществляется путем нажатия на нужную карточку и передвижения ее в левую сторону.

### 3.2.3 Экран выбора игры

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описание

Рисунок 7 - Экран выбора игры

На странице выбора игры представлены три игры, которые пользователь может выбрать для игры:

1. "FindRight Word" — это игра, в которой пользователю предстоит найти правильный перевод слова из четырех доступных вариантов перевода. Игроку будет предложено слово на одном языке, и он должен выбрать правильный перевод из представленных вариантов.

2. "Enter Right Word" — это игра, в которой пользователю нужно ввести правильный перевод слова в поле ввода, используя экранную клавиатуру. Игроку будет показано слово на одном языке, и он должен правильно ввести его перевод в поле ввода.

3. "Enter Right Word" — это игра, в которой пользователю нужно прослушать слово, а затем выбрать правильный перевод из 4 возможных вариантов перевода.

Все игры предлагают интерактивный и увлекательный способ тренировки и проверки знания переводов слов. Пользователь может выбрать любую из игр в зависимости от своих предпочтений и целей обучения.

### 3.2.4 Экран выбора папок, которые будут участвовать в игре

Изображение выглядит как текст

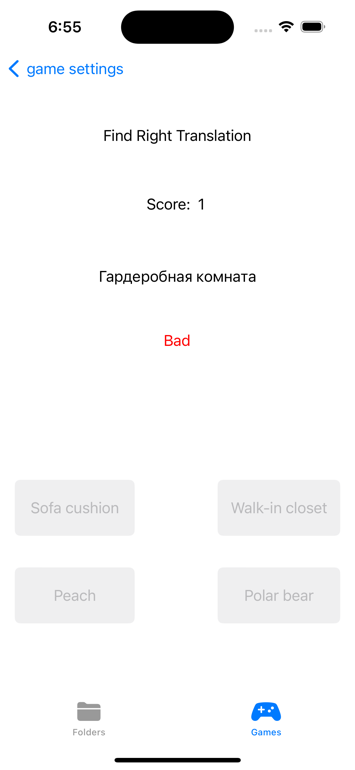
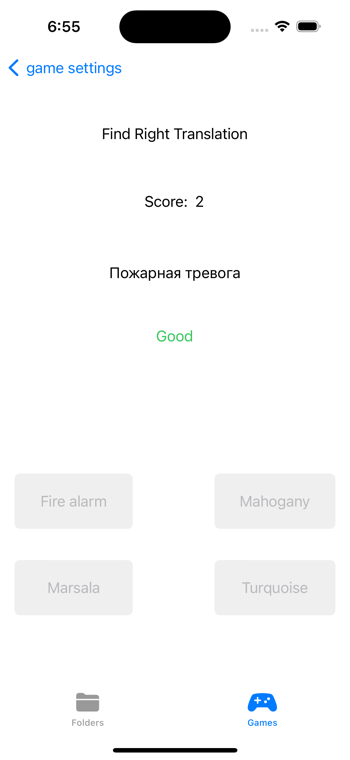
Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 8 - Экран выбора папок, которые будут участвовать в игре

Страница с выбором папок для участия в игре представляет собой список всех созданных пользователем папок. Чтобы выбрать папки, карточки которых будут использоваться в игре, нужно нажать на желаемую папку. Напротив выбранной папки появится значок с галочкой, показывающий, что эта папка добавлена в список выбранных. Обратите внимание, что кнопка "Go!" будет неактивна, пока общее количество выбранных карточек не превысит 4 штук. Выбранные папки могут быть отменены путем повторного нажатия на них, что уберет галочку и удалит папку из списка выбранных.

### 3.2.5 Экран игры (Поиск правильного перевода)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 - Экран игры

Игра "Поиск правильного перевода" представляет собой интерактивное приложение, в котором пользователю предстоит выбрать правильный перевод слова или выражения из четырех возможных вариантов. Если пользователь выбирает верный вариант перевода, ему начисляется одно игровое очко, а на экране появляется зеленая надпись "Good". В случае, если пользователь выбирает неверный перевод, он не получает игровых очков, и на экране появляется красная надпись "Bad".

Эта игра позволяет пользователям улучшить свой словарный запас и развивать навыки перевода. Кроме того, игра способствует развитию быстроты реакции и внимания.

### 3.2.6 Экран игры (Написать правильный перевод)

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Рисунок 10 - Экран игры

Игра "Ввод правильного перевода" предлагает пользователю ввести правильный перевод слова или выражения в поле для ввода и нажать кнопку "Check". Если пользователь вводит верный перевод, ему начисляется одно игровое очко, а на экране появляется зеленая надпись "Good". В случае, если пользователь вводит неверный перевод, он не получает игровых очков, и на экране появляется красная надпись "Bad".

Эта игра помогает пользователям оттачивать свои навыки перевода и проверять свои знания. Она способствует развитию точности и внимания к деталям, поскольку требует правильного ввода перевода. Кроме того, игра может быть интересным способом улучшить словарный запас и практиковать иностранный язык.

### 3.2.7 Экран игры (Поиск правильного перевода по аудио)

Изображение выглядит как текст, снимок экрана, дизайн

Автоматически созданное описаниеИзображение выглядит как текст, дизайн, снимок экрана

Автоматически созданное описаниеИзображение выглядит как текст, дизайн, снимок экрана

Автоматически созданное описание

Рисунок 11 - Экран игры

Игра "Поиск правильного перевода по аудио" представляет собой интерактивную задачу, в которой пользователь должен прослушать аудио запись слова или фразы, а затем выбрать правильный перевод из четырех вариантов ответа. Для начала игры пользователь нажимает на кнопку "Воспроизвести аудио", после чего проигрывается запись. После прослушивания пользователю предлагается выбрать один из четырех вариантов перевода, нажав на соответствующую кнопку.

Если пользователь выбирает правильный перевод, ему начисляется одно игровое очко, а на экране появляется сообщение " Good " с зеленым цветом. В случае, если пользователь выбирает неправильный перевод, он не получает игровых очков, и на экране появляется сообщение " Bad " с красным цветом.

Эта игра создана для того, чтобы сделать изучение иностранного языка более интерактивным и увлекательным, а также помочь пользователям достичь более высокого уровня владения языком.

### 3.2.8 Экран окончания игры

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 - Экран окончания игры

На странице, на которую пользователь попадает автоматически после окончания любой игры, отображается информация о количестве очков, которые пользователь набрал, а также максимальное количество очков, которое он может набрать. Это позволяет пользователю видеть свой прогресс и стремиться к достижению максимального результата.

Дополнительно, на странице расположена кнопка, которая при нажатии перенаправляет пользователя на экран с выбором игр. Это предоставляет пользователю возможность сразу начать новую игру или выбрать другую игру для игры. Кнопка обеспечивает удобный и быстрый доступ к выбору игр и продолжению игрового процесса.

Таким образом, страница после окончания игры предоставляет пользователю обзор его результатов, возможность перейти к выбору новой игры и поддерживает его мотивацию и вовлеченность в игровой процесс.

# Заключение

В ходе разработки и создания мобильного приложения для изучения иностранных языков были успешно выполнены поставленные задачи. Была разработана полнофункциональная система, основанная на языке программирования Swift и паттерне проектирования Model-View-Controller (MVC), которая позволяет пользователям эффективно изучать и запоминать новые слова и фразы.

Одной из основных задач приложения было создание системы хранения и организации папок с карточками для изучения слов. С помощью технологии FileManager была обеспечена надежность хранения данных и возможность легкого добавления, удаления и сортировки карточек внутри папок. Это дает пользователям возможность систематизировать свой контент и настраивать его по своим потребностям.

Также были реализованы игровые механики, которые позволяют пользователям проверять и улучшать свои знания языка. Встроенные игры предлагают интерактивный и увлекательный подход к обучению, что способствует повышению мотивации и результативности изучения языков.

Важным аспектом разработки приложения было создание удобного пользовательского интерфейса. С использованием фреймворка UIKit были реализованы интуитивно понятные и привлекательные компоненты интерфейса, которые обеспечивают легкую навигацию по приложению и удобное взаимодействие с его функциональными возможностями.

В результате выполнения поставленных задач было разработано мобильное приложение, которое позволяет пользователям эффективно изучать иностранные языки, создавать персональные папки с карточками и проверять свои знания с помощью встроенных игр. Реализация приложения на языке Swift с применением паттерна MVC, FileManager и UIKit обеспечивает надежность, функциональность и удобство использования. Приложение представляет собой полезный инструмент для всех, кто стремится к эффективному изучению иностранных языков.

# Список использованных источников

1. Apple Developer Documentation [Электронный ресурс]. – Режим доступа: <https://developer.apple.com/documentation/> – Заглавие с экрана.
2. Swift.org Documentation [Электронный ресурс]. – Режим доступа: <https://docs.swift.org/> – Заглавие с экрана.
3. UIKit Framework Documentation [Электронный ресурс]. – Режим доступа: <https://developer.apple.com/documentation/uikit/> – Заглавие с экрана.
4. FileManagerDocumentation [Электронный ресурс]. – Режим доступа: <https://developer.apple.com/documentation/foundation/filemanager> – Заглавие с экрана.
5. iOS приложение на Swift [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=lSyo8Yxl8iI&list=PLY4rE9dstrJyPbNmc1GjVUrfKjRakp-Dj. – Заглавие с экрана.
6. Navigationcontroller, segue [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=x2iknlTdx-w&t=456s. – Заглавие с экрана.

# Приложения

**Приложение 1**

**Программный код структуры данных и системы хранения**

//

// DataStorage.swift

// App3

//

// Created by Никита Пеканов on 18.04.2023.

//

import Foundation

// Структура ячейки, хранящей слово и его перевод

struct Cell: Codable {

var word: String

var translation: String

}

// Структура папки, содержащей массив ячеек и методы для работы с ними

struct Folder: Codable, Equatable{

var name : String

var cells: [Cell]

mutating funcaddCell(newCell: Cell) {

cells.append(newCell)

}

mutating funcremoveCell(at index: Int) {

cells.remove(at: index)

}

static func ==(lhs: Folder, rhs: Folder) -> Bool {

return lhs.name == rhs.name

}

}

// Структура, содержащая массив папок и методы для работы с ними

struct DataStorage: Codable{

static var shared = DataStorage()

var folders: [Folder] = []

mutating funcfolderSort() {

folders.sort {$0.name.localizedCaseInsensitiveCompare($1.name) == .orderedAscending}

}

mutating funccellSort(folderIndex: Int) {

folders[folderIndex].cells.sort{$0.word.localizedCaseInsensitiveCompare($1.word) == .orderedAscending}

}

mutating funcaddFolder(\_ folder: Folder) {

folders.append(folder)

print("addFolder",folders.count)

saveData()

}

mutating funcremoveFolder(at index: Int) {

folders.remove(at: index)

saveData()

}

mutating funcgetFolder(at index: Int) -> Folder? {

guard folders.indices.contains(index) else {

return nil

}

return folders[index]

}

mutating funcupdateFolder(at index: Int, with newFolder: Folder) {

folders[index] = newFolder

saveData()

}

// Сохранениеданных в файл

funcsaveData() {

let encoder = JSONEncoder()

encoder.outputFormatting = .prettyPrinted

do {

let data = try encoder.encode(self.folders)

try data.write(to: dataFilePath(), options: .atomic)

} catch {

print("Failed to save data: \(error.localizedDescription)")

}

}

// Загрузкаданныхизфайла

mutating funcloadData() {

let decoder = JSONDecoder()

do {

let data = try Data(contentsOf: dataFilePath())

self.folders = try decoder.decode([Folder].self, from: data)

} catch {

print("Failed to load data: \(error.localizedDescription)")

}

}

// Путь к файлу с данными

private funcdataFilePath() -> URL {

let documentsURL = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first!

return documentsURL.appendingPathComponent("data.json")

}

funccreateDataFile() {

let documentsDirectory = FileManager.default.urls(for: .documentDirectory, in: .userDomainMask).first!

let fileURL = documentsDirectory.appendingPathComponent("data.json")

if !FileManager.default.fileExists(atPath: fileURL.path) {

do {

let data = try JSONEncoder().encode(DataStorage.shared)

try data.write(to: fileURL, options: .atomic)

} catch {

print("Error creating file: \(error.localizedDescription)")

}

}

}

}

**Приложение 2**

**Программный код абстрактной игры**

//

// GameAbstract.swift

// App3

//

// Created by Никита Пеканов on 23.04.2023.

//

import Foundation

class GameAbstract {

var allCellsForGame: [Cell]?

var maxScore: Int?

var score: Int

init(folders: [Folder]) {

self.score = 0

allCellsForGame = getAllCells(folders: folders)

maxScore = allCellsForGame?.count

shuffle()

}

funcgetAllCells(folders: [Folder]) -> [Cell] {

return folders.flatMap { $0.cells }

}

funcshuffle() {

allCellsForGame?.shuffle()

}

funcgetFirstCell() -> Cell? {

return allCellsForGame?.first

}

funcscoreUp() {

score += 1

}

funcgameStep(enteredWord: String) -> Bool {

returntrue

}

}

**Приложение 3**

**Программный код игры 1**

**"Поиск правильного перевода"**

//

// GameEnterRightWord.swift

// App3

//

// Created by Никита Пеканов on 23.04.2023.

//

import Foundation

class GameEnterRightWord: GameAbstract {

override init(folders: [Folder]) {

super.init(folders: folders)

}

override funcgameStep(enteredWord: String) -> Bool {

if getFirstCell()?.translation.lowercased() == enteredWord {

scoreUp()

allCellsForGame?.removeFirst()

return true

} else {

allCellsForGame?.removeFirst()

returnfalse

}

}

}

**Приложение 4**

**Программный код игры 2**

**"Написать правильный перевод"**

//

// GameFindRightWord.swift

// App3

//

// Created by Никита Пеканов on 28.04.2023.

//

import Foundation

class GameFindRightWord: GameAbstract {

var indexOfSelectedCell: Int = 0

override init(folders: [Folder]) {

super.init(folders: folders)

}

funcgetTranslationsArray() -> [String]? {

guard let cells = allCellsForGame, cells.count> 3 else {

return nil

}

var arr: [String] = []

if let firstTranslation = allCellsForGame?[indexOfSelectedCell].translation {

arr.append(firstTranslation)

}

while arr.count<= 3 {

if let randomTranslation = allCellsForGame?.randomElement()?.translation, !arr.contains(randomTranslation) {

arr.append(randomTranslation)

}

}

arr.shuffle()

return arr

}

override funcgameStep(enteredWord: String) -> Bool {

if allCellsForGame?[indexOfSelectedCell].translation.lowercased() == enteredWord {

scoreUp()

indexOfSelectedCell += 1

return true

} else {

indexOfSelectedCell += 1

returnfalse

}

}

}

**Приложение 5**

**Программный код игры 3**

**"Поиск правильного перевода по аудио"**

//

// Audiotranslation.swift

// App3

//

// Created by Никита Пеканов on 21.05.2023.

//

import Foundation

class GameAudioTranslation: GameAbstract {

var indexOfSelectedCell: Int = 0

override init(folders: [Folder]) {

super.init(folders: folders)

}

funcgetTranslationsArray() -> [String]? {

guard let cells = allCellsForGame, cells.count> 3 else {

return nil

}

var arr: [String] = []

if let firstTranslation = allCellsForGame?[indexOfSelectedCell].translation {

arr.append(firstTranslation)

}

while arr.count<= 3 {

if let randomTranslation = allCellsForGame?.randomElement()?.translation, !arr.contains(randomTranslation) {

arr.append(randomTranslation)

}

}

arr.shuffle()

return arr

}

override funcgameStep(enteredWord: String) -> Bool {

if allCellsForGame?[indexOfSelectedCell].translation.lowercased() == enteredWord {

scoreUp()

indexOfSelectedCell += 1

return true

} else {

indexOfSelectedCell += 1

returnfalse

}

}

}