**Кыргызско-Российский Славянский университет**

**Кафедра информационных и вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**по дисциплине «РАЗРАБОТКА КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ»**

**Тема: *«Сайт для запоминания информации при помощи карточек»***

**Выполнил: студент группы ЕПИ-2-20 Мирлан уулу Алинур.**

**Руководитель: профессор В.В.Гайдамако**

Оценка:

Дата:

Бишкек 2023

Оглавление

[1 Введение 3](#_Toc124406352)

[2 Разработка программы 4](#_Toc124406353)

[2.1 Технологии разработки. 4](#_Toc124406354)

[2.2 Дизайн и структура 6](#_Toc124406355)

[2.3 Модель данных и алгоритм решения задачи. 11](#_Toc124406356)

[3 Руководство программиста 13](#_Toc124406357)

[4 Руководство пользователя 14](#_Toc124406358)

[5 Выводы 15](#_Toc124406359)

# Введение

Мне, как человеку, заинтересованному иностранными языками, важно не забывать пополнять словарный запас. Вести словарик можно разными способами, будь то запись выражений в специально определенной для словаря группе в мессенджере (как делал я), либо в старой доброй тетради; но какой бы способ это ни был, всё сводится к одному: повтор записей для дальнейшего их запоминания; а делать это вручную представляет большие трудности.

Я решил взяться за автоматизацию процесса слежения за повторениями записей и создать сайт для их хранения и запоминания. Записями являются любая текстовая информация что вы желаете запомнить (в моём случае это иностранные слова). Все записи организованы в тематические колоды, что помогает навести порядок.

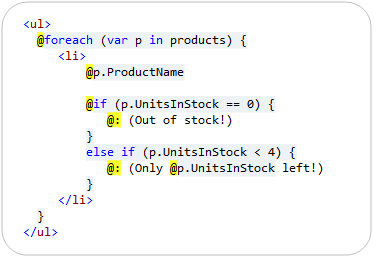
# Разработка программы

## Технологии разработки.



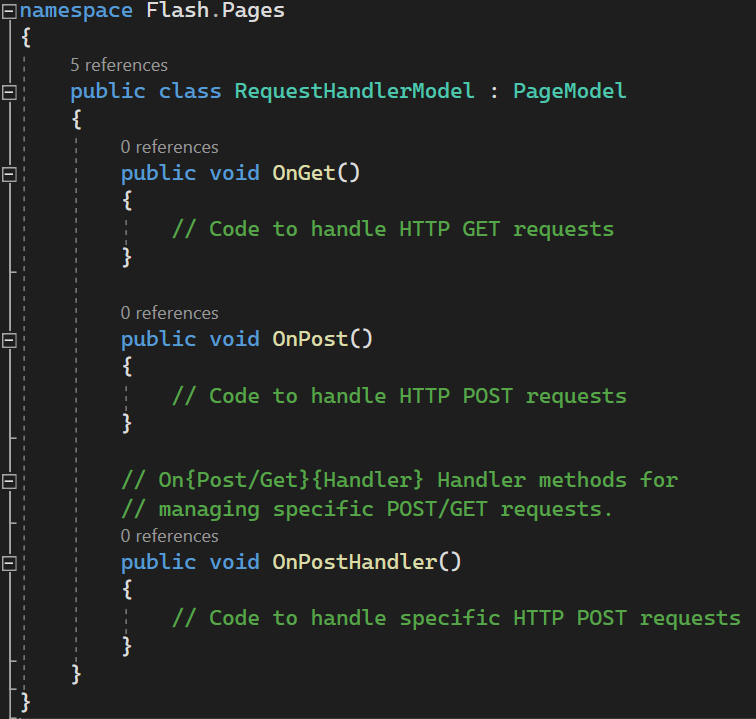
Сердцем моего проекта стал фреймворк для создания динамических веб-страниц, собрат ASP.NET Core MVC **—** ASP.NET Core Razor Pages разработанный Microsoft.

ASP.NET Core Razor Pages построен на основе Razor View Engine, что позволяет разработчикам создавать веб-страницы, используя комбинацию синтаксиса гипертекстовой разметки HTML, языка программирования C#, и Razor.



*Рис 1. Пример использования синтаксиса Razor для внедрения кода C# в гипертекстовую разметку HTML.*

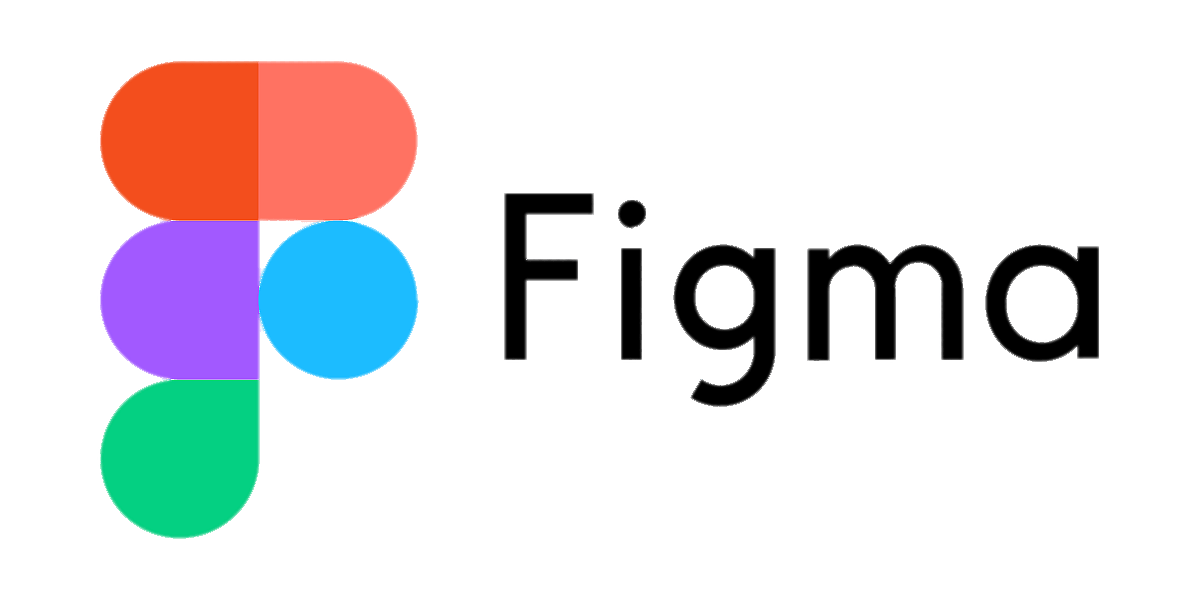
Каждая веб-страница Razor оснащена классом “обработчиком” C# которая отвечает на HTTP-запросы. Когда для страницы делается запрос, платформа использует URL-адрес запроса, чтобы перенаправить его к соответствующему классу обработчику, который, в свою очередь, вызывает подходящий метод для выведения результата **—** веб-страницы.



*Рис 2. Пример типичного Razor Page класса с методами для обработки HTTP запросов.*

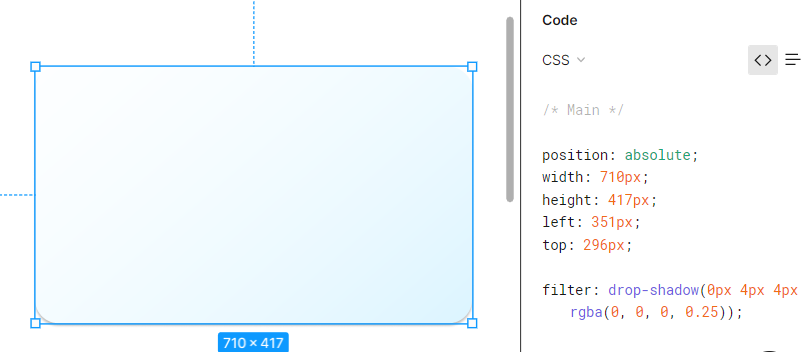
## Дизайн и структура

Основополагающими технологиями для создания веб-страниц служат HTML (Hyper Text Markup Language) **—** структура страницы, CSS (Cascading Style Sheets) **—** внешний вид структуры страницы, и JavaScript **—** поведение; каждый из которых был использован в процессе разработки веб-приложения.



Для воплощения предварительного дизайна веб-сайта использовалась программа Figma **—** инструмент для проектирования, основанный на облачных технологиях.

Благодаря Figma процесс вёрстки веб-сайтов намного упрощается, поскольку большинство того, что создается при помощи приложения уже имеет CSS стили, наложенные на них, что позволяет разработчикам перенести их на свой проект.



*Рис 3. Figma отображает стили, наложенные на элемент.*

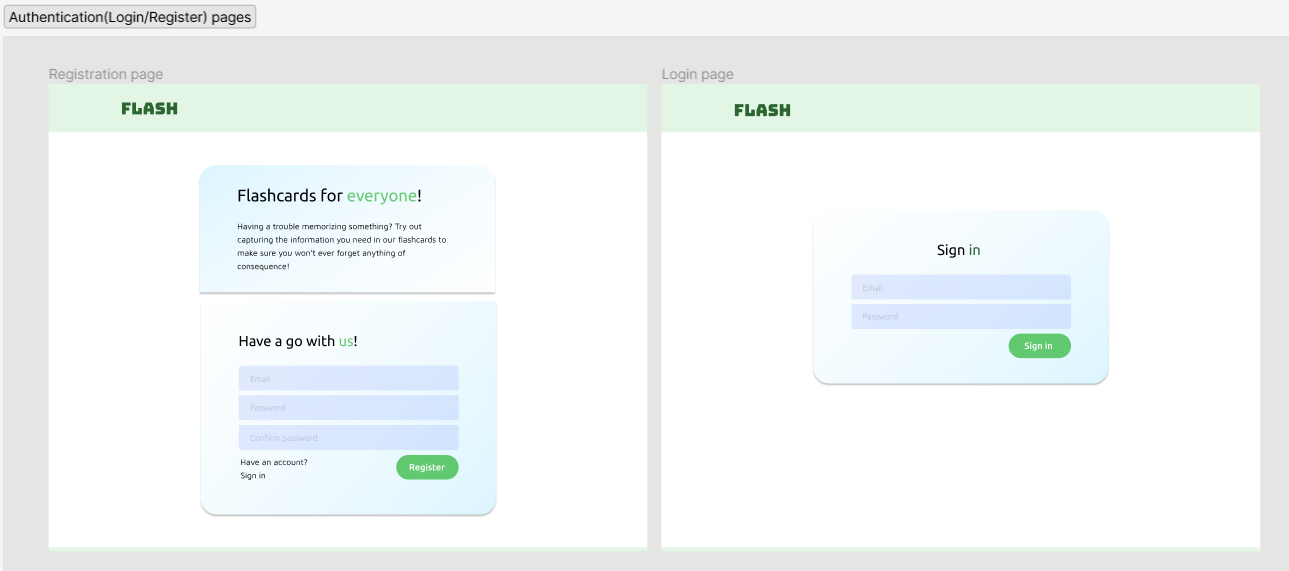
Для моего веб-сайта потребовалось создать 8 веб-страниц, со следующей иерархической структурой:

* Identity
  + - 1. Login
      2. Register
* Card

1. Add
2. Edit
3. Browse
4. Decks
5. Rehearse
6. Settings

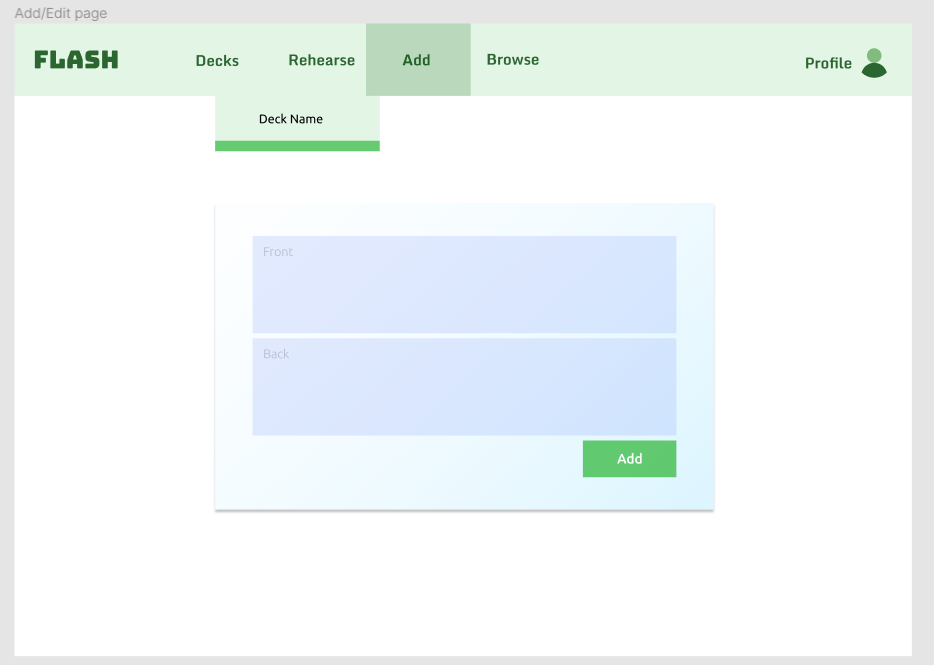
Пройдемся по каждой из них:

1. Register, 2. Login**—** точки регистрации и авторизации соответственно.



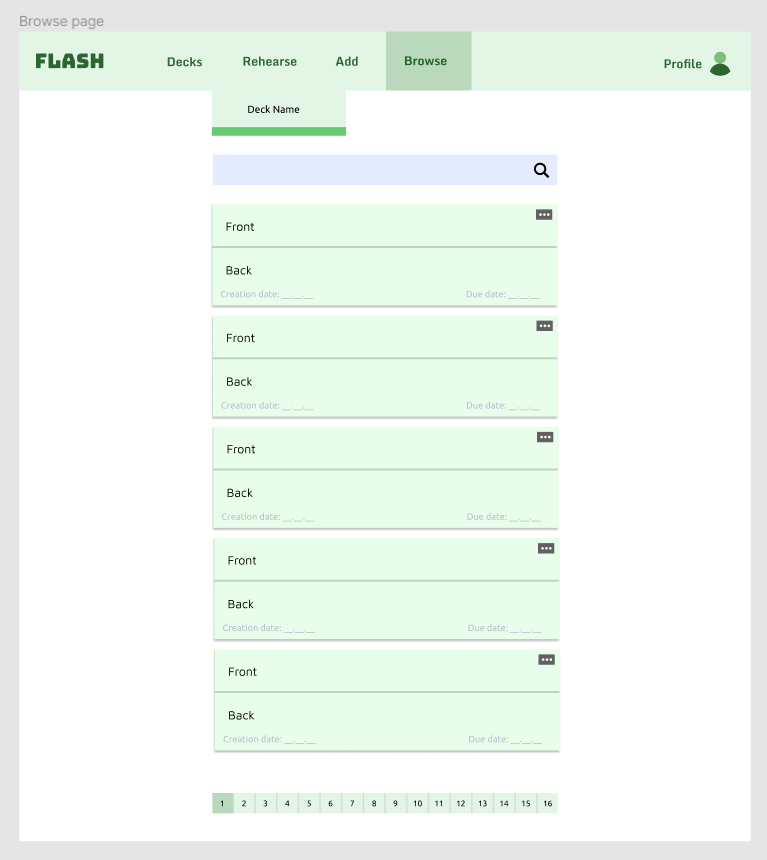
*Рис 4. Дизайн страниц Register и Login.*

3. Add, 4. Edit **—** точки добавления и редактирования карточек в соответствующие колоды.



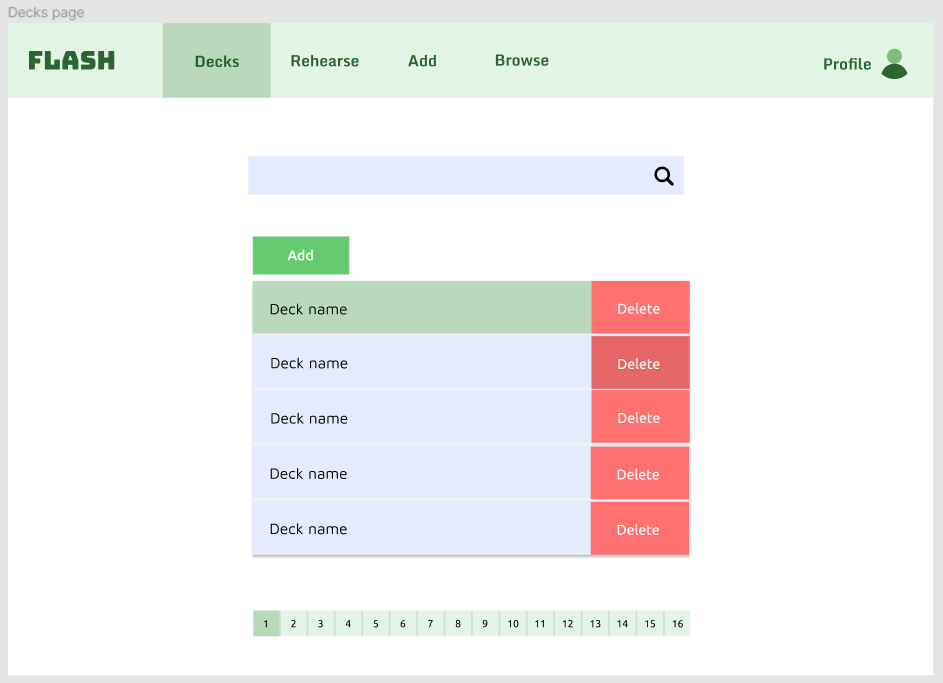
*Рис 5. Дизайн страниц Add и Edit.*

5. Browse **—** точка отображения всех карточек данной колоды.



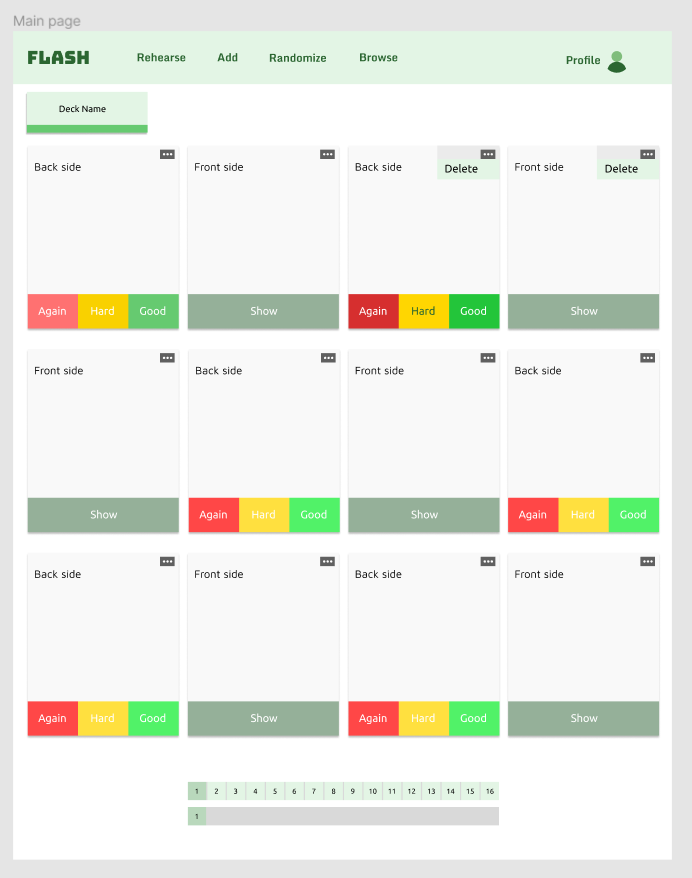
*Рис 6. Дизайн страницы Browse.*

6. Decks **—** точка отображения, добавления, удаления и выборки колоды.



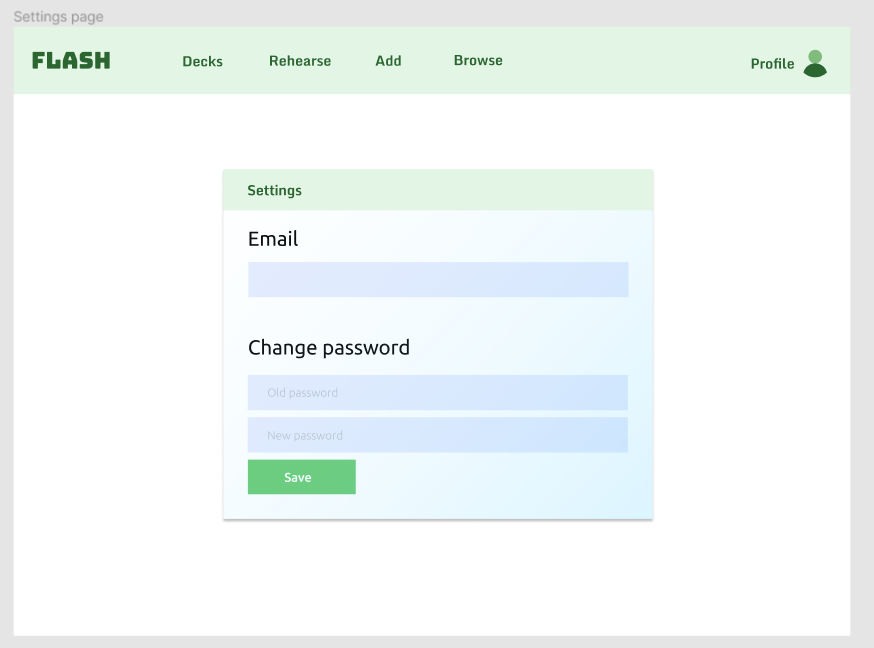
*Рис 7. Дизайн страницы Decks.*

7. Rehearse **—** главная страница веб-сайта, отображающая набор карточек, которые необходимо повторить.



*Рис 8. Дизайн страницы Rehearse.*

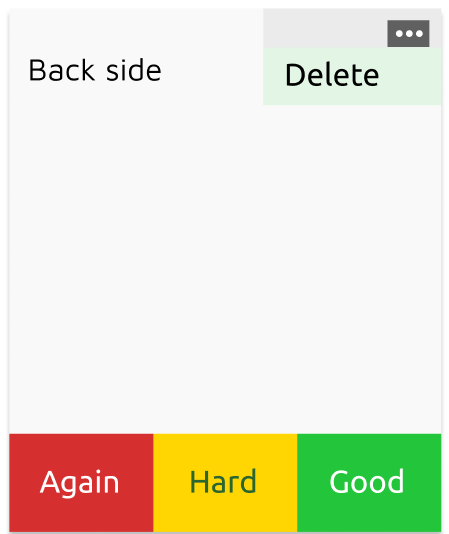
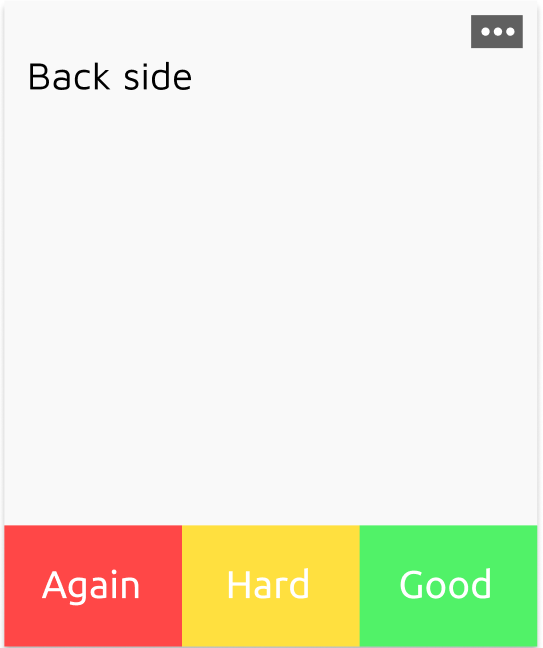
8. Settings **—** точка смены пароля и отображения почты.



*Рис 9. Дизайн страницы Settings*

Интервал повторения карт зависит от того, насколько легко было вспомнить карту пользователю. Поэтому карточки содержат кнопки, соответствующие трём уровням сложности: Again, Hard и Good.

У каждой карточки (будь то на странице Rehearse или Browse) также присутствует кнопка для манипуляции над картой.

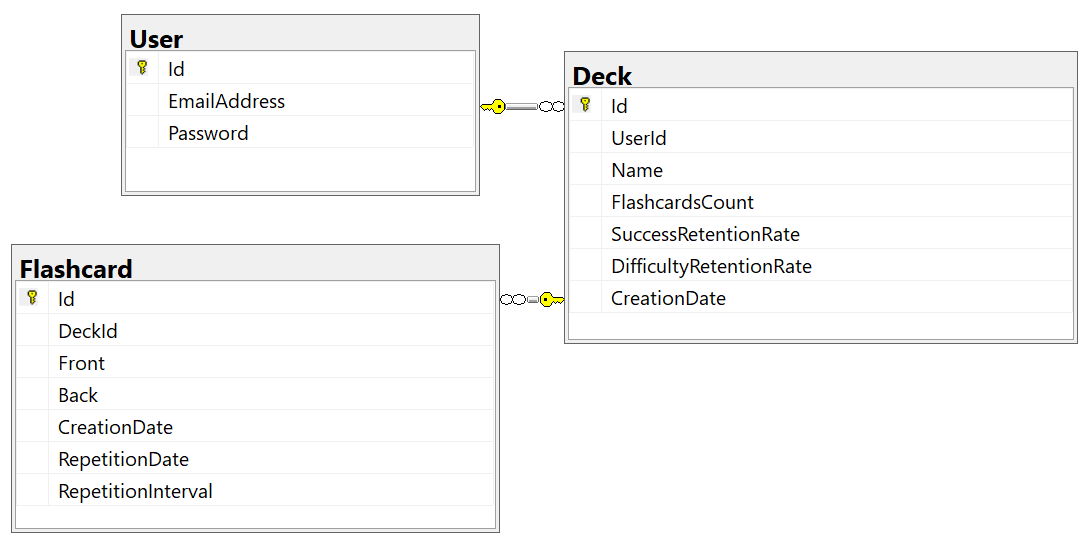
 

*Рис 10. Дизайн карточки с раскрытым и закрытым функционалом.*

## Модель данных и алгоритм решения задачи.

Для хранения информации была использована реляционная база данных SQL Server с использованием языка T-SQL (Transact Structured Query Language).

Модель данных включает пользователей, их колоды, и карточки в колодах.



*Рис 11. Диаграмма базы данных SQL Server*

Алгоритм определения следующей даты повторения карточек можно описать следующим псевдокодом:

SuccessRetentionRate = .85

DifficultyRetentionRate = 1.2

RepetitionInterval

If pressend ‘Again’: RepetitionInterval = 0

If RepeatInterval is 0:

If pressed ‘Hard’: RepetitionInterval = 0

If pressed ‘Good’: RepetitionInterval = 2

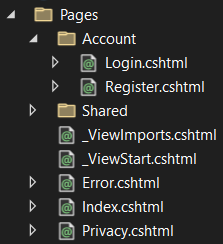
Else

If pressed ‘Hard’: RepetitionInterval = RepetitionInterval \* DifficultyRetentionRate

If pressed ‘Good’: RepetitionInterval = RepetitionInterval + (RetentionRate \* RepeatInterval)

# Руководство программиста

В ASP.NET Core Razor Pages маршрутизация по веб-сайту осуществляется за счёт иерархии папок и файлов в проекте.



*Рис 12. Пример маршрутизации*

В примере выше адреса страниц будут следующими:

* Login – /Account/Login
* Register – /Account/Register
* Error – /Error
* Index – /Index
* Privacy – /Privacy

# Руководство пользователя

Приветствую вас, Пользователь! При первом доступе на веб-сайт, вас попросят зарегистрироваться в целях идентификации, после чего вы получите доступ к веб-сайту.

Прежде чем добавлять карточки, вам нужно создать колоду для её хранения. Сделано это с целью разделения разных по тематике карточек, скажем, есть у вас одна колода для хранения иностранных слов, а вторая для тем по подготовке к экзамену.

Все карточки, которые необходимо повторить, отображаются на странице Rehearse. Вы всегда можете изменить, либо удалить карту, при помощи кнопки дополнительного функционала справа сверху карточек.

Если же вы хотите просмотреть все существующие карточки колоды, перейдите на страницу Browse, где вы также можете манипулировать картами с помощью кнопки

В целях удобства, на страницах Decks и Browse есть поля для поиска желаемой вами информации. В странице Decks вы можете запросить поиск колоды с определенным именем, а в Browse карточку, содержащую заданный текст.

Для смены пароля или доступа к почте перейдите на страницу Profile -> Settings. И наконец, для выхода из сессии, пройдите на Profile -> Logout.

# Выводы

Работа над данным проектом стала для меня большой ступенью в развитии представления о процессе разработки веб-приложения, о структуризации программы и упрощении кода.

Используемая литература

BGI Documentation: <https://home.cs.colorado.edu/~main/bgi/doc/>

Website StackOverflow: <https://stackoverflow.com/>