Описание проекта

Содержание

1. Описание физической архитектуры системы
    - Физическая архитектура
    - Структура проекта
2. Описание даталогической архитектуры системы
    - ER-диаграмма
    - Описание сущностей и атрибутов
        - Задача
        - Занятие
        - Заметка
        - Ресурс
    - Описание связей между сущностями
3. Описание программы. Описание основных функций каждого раздела.
    - Расписание
    - Задачи, планировщик
    - Заметки
    - Ресурсы

1. Описание физической архитектуры системы

Физическая архитектура

Система представляет собой мобильное приложение, разработанное на языке программирования Dart версии 3.3.4 с использованием фреймворка Flutter версии 3.19.6. Приложение работает на операционной системе Android 6 и выше, а также на iOS старше 12 версии. Для хранения данных используется локальная база данных SQLite3, взаимодействие с которой осуществляется через пакет drift версии 2.17.0.

Для разработки был выбран фреймворк Flutter, так как он позволяет создавать кроссплатформенные мобильные приложения, а это позволяет сэкономить время и ресурсы на разработку и поддержку приложения. Фреймворк позволяет создавать красивые и быстрые приложения, которые могут работать на разных устройствах и операционных системах, таких как Android, iOS, Windows, macOS и Linux без изменения кода. Также Flutter имеет большое сообщество разработчиков, что позволяет быстро найти ответы на вопросы и решения проблем, а также использовать готовые решения и пакеты, что ускоряет разработку. Еще одним преимуществом Flutter является то, что он разрабатывается и поддерживается компанией Google, что гарантирует его актуальность и поддержку в будущем.

SQLite3 была выбрана в качестве базы данных, так как она является легковесной и простой в использовании, что позволяет быстро и легко создавать и изменять базу данных. К тому же данная база данных позволяет хранить данные локально на устройстве пользователя, что обеспечивает быстрый доступ к данным и работу приложения без подключения к интернету. Также SQLite3 поддерживается пакетом drift, который позволяет работать с базой данных на языке Dart, что

упрощает взаимодействие с базой данных и позволяет использовать привычный язык программирования.

В рамках текущей реализации было решено разработать полностью автономное приложение, не требующее подключения к интернету. Все данные хранятся локально на устройстве пользователя. Такой выбор обоснован тем, что приложение расчитано на использование одним пользователем и не предполагает совместной работы нескольких пользователей. В дальнейшем планируется коммерциализация приложения путем добавления возможности синхронизации данных с облачным хранилищем, что позволит пользователю иметь доступ к своим данным с разных устройств. Также это позволит реализовать возможность делиться данными с другими пользователями.

Структура проекта

Проект разделен на несколько основных частей:

- `lib` - основная директория, в которой находятся все файлы приложения
- `lib/src/` - директория, в которой находятся точка входа в приложение и файлы, отвечающие за инициализацию приложения
- `lib/src/common` - директория, в которой находятся общие для всего приложения файлы, а точнее файлы, содержащие общие константы, стили, виджеты, настройки маршрутизации и т.д.
- `lib/src/feature` - директория, в которой находятся файлы, отвечающие за отдельные части приложения
- `lib/src/feature/{feature_name}` - директория, в которой находятся файлы, отвечающие за возможности приложения, связанные с определенной областью жизни пользователя.

В нашем случае приложение разделено на 4 основные возможности:

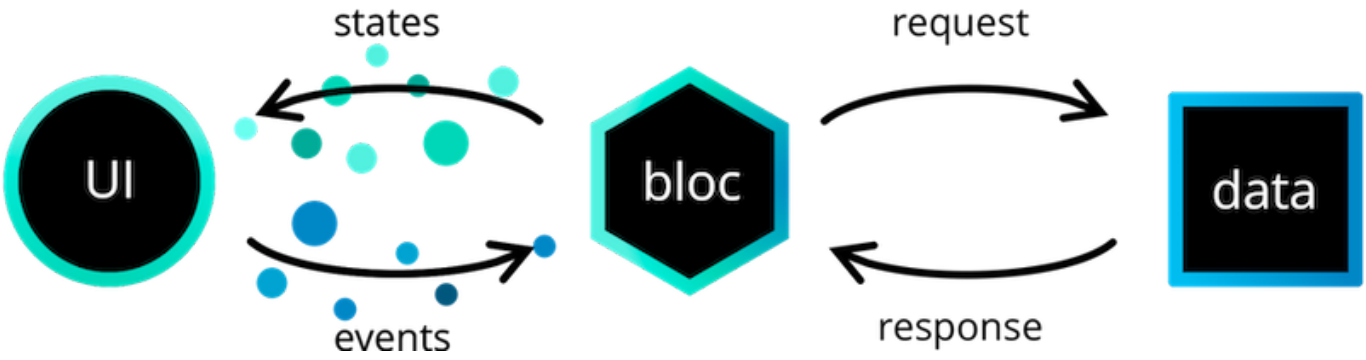
- `lib/src/feature/tasks` - директория, в которой находятся файлы, отвечающие за отображение списка задач, т.ч. добавление и редактирование задач, отображение деталей задачи, фильтрация задач по статусу и приоритету, удаление задач, взятие задачи в работу, завершение задачи, а также самое основное - запуск алгоритма вычисления важности задач, позволяющего определить и предложить к решению наиболее важные на текущий момент задачи, что поможет пользователю оптимизировать свое время, повысить производительность и выполнить все задачи в срок.
- `lib/src/feature/calendar` - директория, в которой находятся файлы, отвечающие за работу с календарем, т.ч. отображение расписания и задач, добавление и редактирование занятий, выбор активной недели.
- `lib/src/feature/notes` - директория, в которой находятся файлы, отвечающие за работу с заметками, т.ч. добавление и редактирование заметок, удаление заметок, добавление в избранное. Заметки позволяют писать текст в markdown, что позволяет пользователю форматировать текст, добавлять ссылки, изображения, списки и т.д.
- `lib/src/feature/resources` - директория, в которой находятся файлы, отвечающие за работу с ресурсами, т.ч. добавление и редактирование ресурсов, удаление ресурсов, добавление в избранное. Ресурсы позволяют пользователю хранить ссылки на веб-страницы, а в будущем планируется добавить возможность хранить файлы, что позволит пользователю хранить важные файлы на устройстве и иметь к ним доступ в любое время.

Каждая feature, как уже было сказано, отвечает за определенную область жизни пользователя. Каждая из них состоит из нескольких файлов, отвечающих за разные части функционала. Например, в feature tasks есть следующие файлы:

- `lib/src/feature/tasks/tasks_screen.dart` - файл, отвечающий за отображение списка задач, а также за добавление и редактирование задач. В этом файле находится виджет, который отображает список задач, а также кнопку добавления новой задачи. При нажатии на кнопку открывается новая страница, на которой пользователь может ввести название задачи, описание, приоритет, крайний срок и т.д. После ввода данных задача сохраняется в базе данных и отображается в списке задач.
- `lib/src/feature/tasks/task_put/task_put_screen.dart` - файл, отвечающий за редактирование задачи. В этом файле находится виджет, который отображает данные задачи, а также кнопку сохранения изменений. При нажатии на кнопку данные задачи обновляются в базе данных и отображаются в списке задач. На этой же странице можно удалить задачу.
- `lib/src/feature/tasks/entities/` - директория, в которой находятся файлы, отвечающие за сущности задач, т.ч. базовая сущность задачи и обычная, содеожащая дополнительные поля, такие как список идентификаторов подзадач и задач, от которых зависит данная задача.

Основной код логики каждой feature находится в файле `lib/src/feature/{feature_name}/bloc/{feature_name}_bloc.dart`, где находится класс, отвечающий за бизнес-логику данной feature. В случае с tasks это класс `TasksBloc`, который отвечает за работу с задачами, а именно за добавление, редактирование, удаление задач, а также за запуск алгоритма вычисления важности задач.

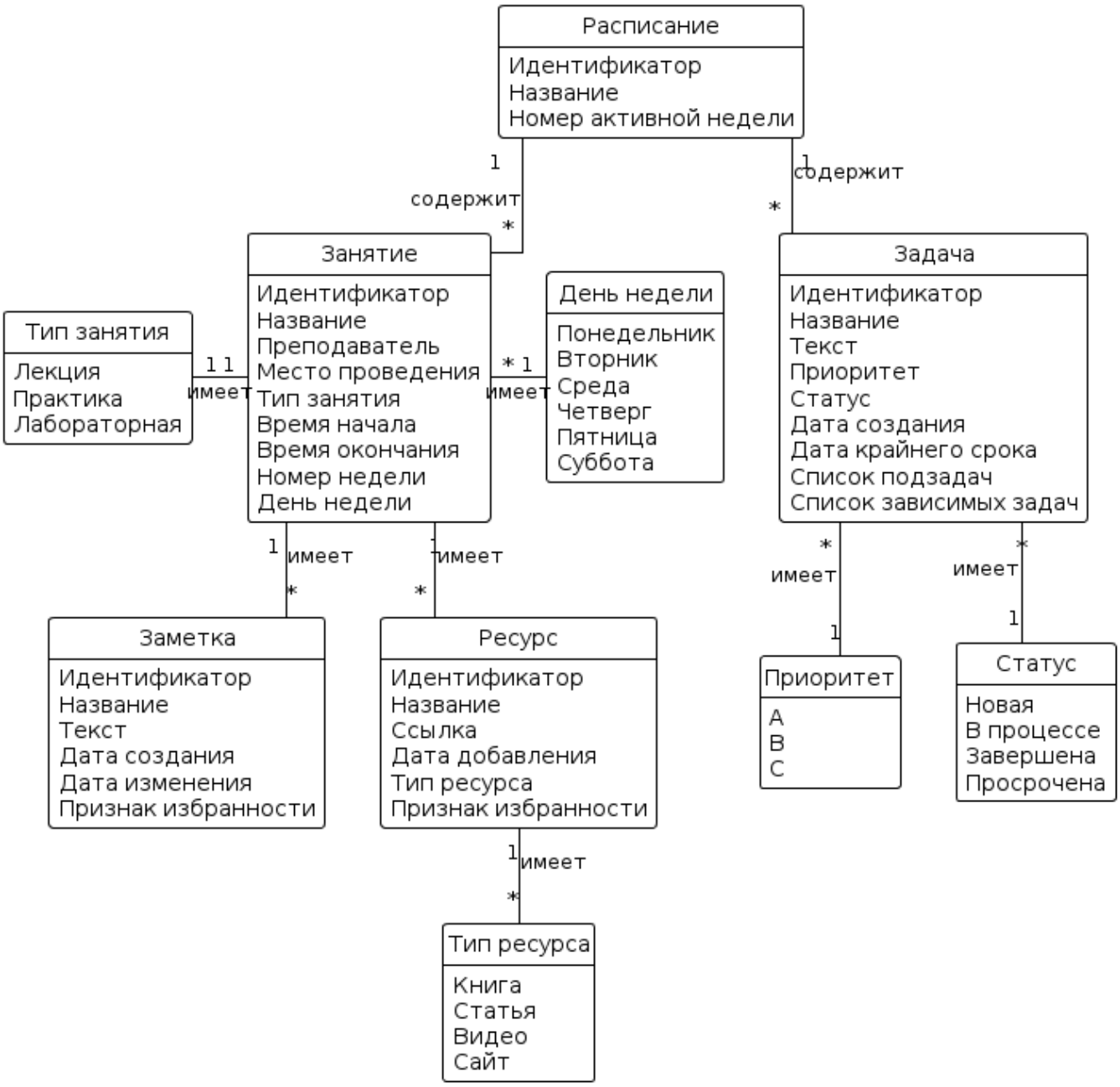
Bloc это паттерн, который позволяет разделить логику приложения на несколько частей, что упрощает разработку и поддержку приложения. В данном случае bloc отвечает за бизнес-логику, а виджеты за отображение данных и взаимодействие с пользователем. Таким образом, bloc отвечает за обработку событий, таких как добавление задачи, редактирование задачи, удаление задачи, запуск алгоритма вычисления важности задач и т.д. Также bloc отвечает за обработку запросов к базе данных, таких как получение списка задач, добавление задачи в базу данных, обновление задачи в базе данных и т.д.



В этом классе TaskBloc находятся методы, отвечающие за обработку событий, таких как добавление задачи, редактирование задачи, удаление задачи, запуск алгоритма вычисления важности задач. Также в этом классе находятся методы, отвечающие за запросы к базе данных, такие как получение списка задач, добавление задачи в базу данных, обновление задачи в базе данных и т.д. Таким образом, данный класс отвечает за бизнес-логику приложения, а также за взаимодействие с базой данных.

2. Описание даталогической архитектуры системы

ER-диаграмма



ER-диаграмма представляет собой схему базы данных, которая показывает связи между сущностями и атрибутами. В данной схеме представлены следующие сущности:

- `Задача` - сущность, отвечающая за задачи. В данной сущности хранятся данные о задаче, такие как название, описание, приоритет, крайний срок, статус, список идентификаторов подзадач и задач, от которых зависит данная задача.
- `Занятие` - сущность, отвечающая за занятия. В данной сущности хранятся данные о занятии, такие как название, описание, дата и время начала и окончания, преподаватель, аудитория, тип занятия, номер недели и день недели.
- `Заметка` - сущность, отвечающая за заметки. В данной сущности хранятся данные о заметке, такие как название, текст, дата и время создания, дата и время последнего изменения, флаг избранное.

- `Ресурс` - сущность, отвечающая за ресурсы. В данной сущности хранятся данные о ресурсе, такие как название, описание, ссылка, дата и время создания, тип ресурса, флаг избранное.
- `Расписание` - сущность, отвечающая за расписание. В данной сущности хранятся данные о расписании, такие как название и номер активной недели.

Описание сущностей и атрибутов

Описание сущностей и атрибутов таким образом, как он представлен на ER-диаграмме и в базе данных.

Задача представляет собой сущность, отвечающую за задачи, которые пользователь хочет выполнить. В данной сущности хранятся следующие атрибуты:

- `id` - идентификатор задачи, уникальный идентификатор задачи.
- `title` - название задачи, строка, содержащая название задачи.
- `description` - описание задачи, строка, содержащая описание задачи.
- `priority` - приоритет задачи, строка, содержащая приоритет задачи.
- `createdDate` - дата и время создания задачи, дата и время, когда задача была создана.
- `deadlineDate` - крайний срок задачи, дата и время, когда задачу нужно выполнить.
- `status` - статус задачи, строка, содержащая статус задачи.

`Priority` и `Status` - это перечисления, которые содержат возможные значения приоритета и статуса задачи соответственно.

Занятие представляет собой сущность, отвечающую за занятия, которые пользователь хочет посетить. В данной сущности хранятся следующие атрибуты:

- `id` - идентификатор занятия, уникальный идентификатор занятия.
- `title` - название занятия, строка, содержащая название занятия.
- `description` - описание занятия, строка, содержащая описание занятия.
- `timeStart` - дата и время начала занятия, дата и время, когда занятие начинается.
- `timeEnd` - дата и время окончания занятия, дата и время, когда занятие заканчивается.
- `teacher` - преподаватель занятия, строка, содержащая имя преподавателя.
- `place` - аудитория занятия, строка, содержащая номер аудитории.
- `type` - тип занятия, строка, содержащая тип занятия.
- `weekNumber` - номер недели, на которой проходит занятие, целое число.
- `dayOfWeek` - день недели, на который проходит занятие, строка, содержащая день недели.

`Type` и `DayOfWeek` - это перечисления, которые содержат возможные значения типа занятия и дня недели соответственно.

Заметка представляет собой сущность, отвечающую за заметки, которые пользователь хочет сохранить. В данной сущности хранятся следующие атрибуты:

- `id` - идентификатор заметки, уникальный идентификатор заметки.
- `title` - название заметки, строка, содержащая название заметки.
- `text` - текст заметки, строка, содержащая текст заметки. Поддерживает markdown.
- `createdDate` - дата и время создания заметки, дата и время, когда заметка была создана.
- `lastModifiedDate` - дата и время последнего изменения заметки, дата и время, когда заметка была изменена.

- `isFavorite` - флаг избранное, булево значение, показывающее, является ли заметка избранной.

Ресурс представляет собой сущность, отвечающую за ресурсы, которые пользователь хочет сохранить. В данной сущности хранятся следующие атрибуты:

- `id` - идентификатор ресурса, уникальный идентификатор ресурса.
- `title` - название ресурса, строка, содержащая название ресурса.
- `description` - описание ресурса, строка, содержащая описание ресурса.
- `url` - ссылка на ресурс, строка, содержащая ссылку на ресурс.
- `createdDate` - дата и время создания ресурса, дата и время, когда ресурс был создан.
- `type` - тип ресурса, строка, содержащая тип ресурса.
- `isFavorite` - флаг избранное, булево значение, показывающее, является ли ресурс избранным.

`Type` - это перечисление, которое содержит возможные значения типа ресурса.

Описание связей между сущностями

Связь задачи с подзадачами реализована через таблицу `task_subtask`, которая содержит два поля: `task_id` и `subtask_id`, которые являются идентификаторами задачи и подзадачи соответственно. Таким образом, если задача зависит от другой задачи, то в таблице `task_subtask` будет создана запись с идентификаторами этих задач.

Связь задачи с задачами, от которых она зависит, реализована через таблицу `task_dependency`, которая содержит два поля: `task_id` и `dependency_id`, которые являются идентификаторами задачи и задачи, от которой зависит данная задача соответственно. Таким образом, если задача зависит от другой задачи, то в таблице `task_dependency` будет создана запись с идентификаторами этих задач.