Лабораторная 6. Reports

> Гроб трепещет, грязная битва.

Цель: реализация многослойной архитектуры.

Предметная область

Требуется реализовать механизм автоматизации создания отчётов о проделанной командой работе за определённый период разработки (далее - спринт). В команде один из сотрудников - тимлид, он составляет итоговый отчёт о проделанной работе команды в конце каждого спринта.

Ход событий:

1. Сотрудник может добавлять новые задачи, вносить изменения в существующие, выполнять их.
2. Сотрудник должен писать отчёт о проделанной работе за каждый спринт. Чтобы это сделать он использует список всех изменений, произведённых с момента создания предыдущего отчёта. В течение спринта сотрудник делает отчёт, прикрепляя к нему выполненные за этот период задачи.
3. Тимлид в конце спринта пишет отчёт за всю команду, просматривая список выполненных задач и отчётов.

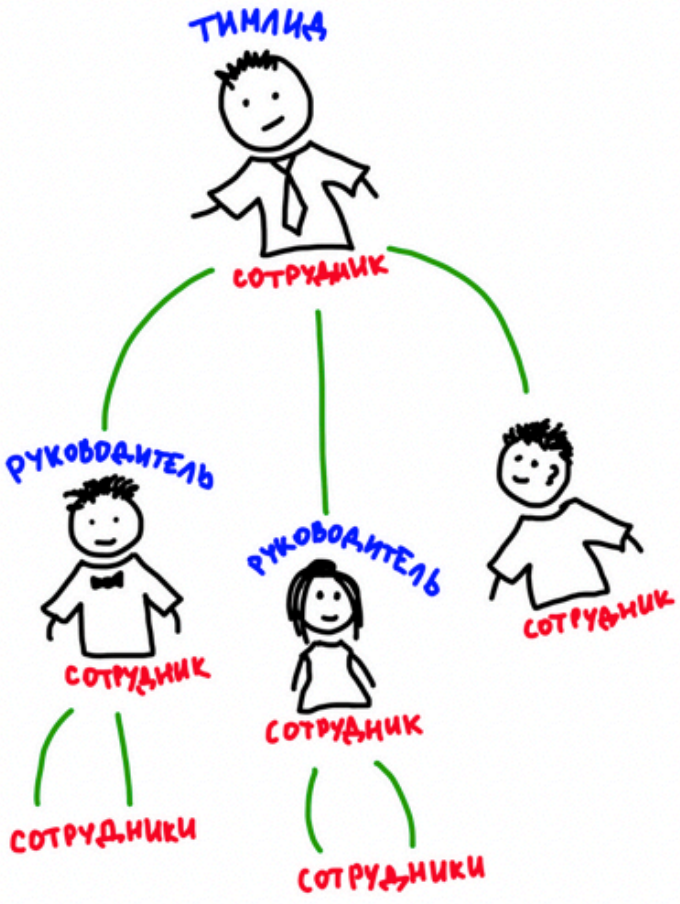
Сущности

- Сотрудник. У сотрудника может быть руководитель и могут быть подчиненные.
- Задача
- Отчёт

В реализованной системе должна быть возможность добавлять новых сотрудников, изменять руководителя сотрудника и получать иерархию всех сотрудников.

Иерархия сотрудников представляется следующим образом:
тимлид - корень дерева, которому подчиняются другие сотрудники и руководители, которые могут иметь или не иметь подчинённых.

Сотрудники могут подчиняться только руководителю.



Задача

Задача может находиться в одном из трёх состояний:

1. Open - задача создана, но к её выполнению ещё не приступили.
2. Active - задача находится в процессе выполнения.
3. Resolved - задача выполнена.

Сотрудник должен иметь возможность добавить комментарий к задаче. Все изменения (состояние/назначенный сотрудник/комментарий) должны фиксироваться во времени. При этом в системе должна быть возможность просмотреть, когда было сделано определённое изменение.

Система управления задачами должна поддерживать следующий функционал:

- поиск задач по ID
- поиск задач по времени создания/последнего изменения
- поиск задач, закреплённых за определённым пользователем (сотрудником команды)
- поиск задач, в которые пользователь вносил изменения
- создание задачи, изменение её состояния, добавления к ней комментария, изменение назначенного за ней сотрудника
- получение списка задач, которые назначены подчинённым определённого сотрудника

Отчёт

Необходимо реализовать возможность написания отчётов за весь спринт. В системе на каждый спринт создается драфт отчёта (то есть черновик, начальный проект отчёта за спринт, который в дальнейшем будет корректироваться), который заполняет каждый сотрудник. Для написания отчёта пользователю системы должна предоставляться возможность получить список всех своих задач за спринт, а также список отчётов своих подчиненных за спринт.

После окончания написания отчёта за спринт сотрудник должен сохранить его в системе. После этого отчёт считается завершённым, доступ на его редактирование закрывается. Тимлид должен иметь возможность видеть статус отчёта. После того, как вся команда загрузит отчёты, тимлид имеет возможность написать общий отчёт за всю команду, который будет агрегировать все остальные.

Интерфейс работы с пользователем

Требуется реализовать такой набор функционала:

- Сотрудники
  - Получение всех сотрудников (с пагинацией и фильтрами)
  - GetById
  - Update
  - Delete

- **Авторизация - выставление, что работа происходит от имени определенного сотрудника (UPD: можно не делать)**
- Задачи
  - GetAll
  - поиск задач по ID
  - поиск задач по времени создания/последнего изменения
  - поиск задач, закреплённых за определённым пользователем (сотрудником команды)
  - поиск задач, в которые пользователь вносил изменения
  - создание задачи, изменение её состояния, добавления к ней комментария, изменение назначенного за ней сотрудника
  - получение списка задач, которые назначены подчинённым определённого сотрудника
  - Добавление задачи, обновление описания задачи
  - Изменение человека, который заасайнен
- Недельный отчет (викли)
  - Создать викли отчет
  - Получить список задач за эту неделю
  - Получить список дейли отчетов подчиненных (для тех, кто написал. Отдельно список тех, кто еще не написал)
  - Добавление задачи в отчет
  - Обновление описания и состояния отчета

Детали реализации

Для этой лабораторной работы в шаблоне нет созданного проекта. Это связано с тем, что студент может выбрать способ реализации данной работы. Пример варианта реализации:

- Три простых проекта:
  - Клиент - консольное приложение, которе содержит логику работы с пользовательским интерфейсом и отправляет запросы на сервер. Может быть реализовано в виде консольного клиента (с или без использования библиотек), либо веб приложения (при большом желании можно что-то сверстать на Blazor\React\Angular\Vue).
  - Сервер - приложение, которое предоставляет API для работы с системой, предоставляет описанный в лабе функционал. Может быть реализован как ASP Web API приложение (которое довольно легко с коробки завести) либо простой TCP сервер, который обрабатывает запросы. Ознакомиться можно тут Tutorial: Create a web API with ASP.NET Core | Microsoft Docs и тут An awesome guide on how to build RESTful APIs with ASP.NET Core | by Evandro Gomes | Medium.
  - Общая сборка. Скорее всего, для общения между клиентом и сервером нужен будет контракт, описание того, какие модели между ними прокидываются. Их нужно выделить в отдельный проект т.к. клиент и сервер должны зависить от этого проекта (и не зависить друг от друга)

Если говорить про Data access layer, то можно:

- написать свою логику хранения данных используя в качестве формата JSON. Но нужно убедиться, что есть возможность после перезагрузки сервера корректно из JSON достать все нужные данные и продолжить работу
- использовать EntityFramework (и подружить его с SQLite) - ORM для работы с базой. Реализовать полноценный слой данных и получить много полезного опыта.