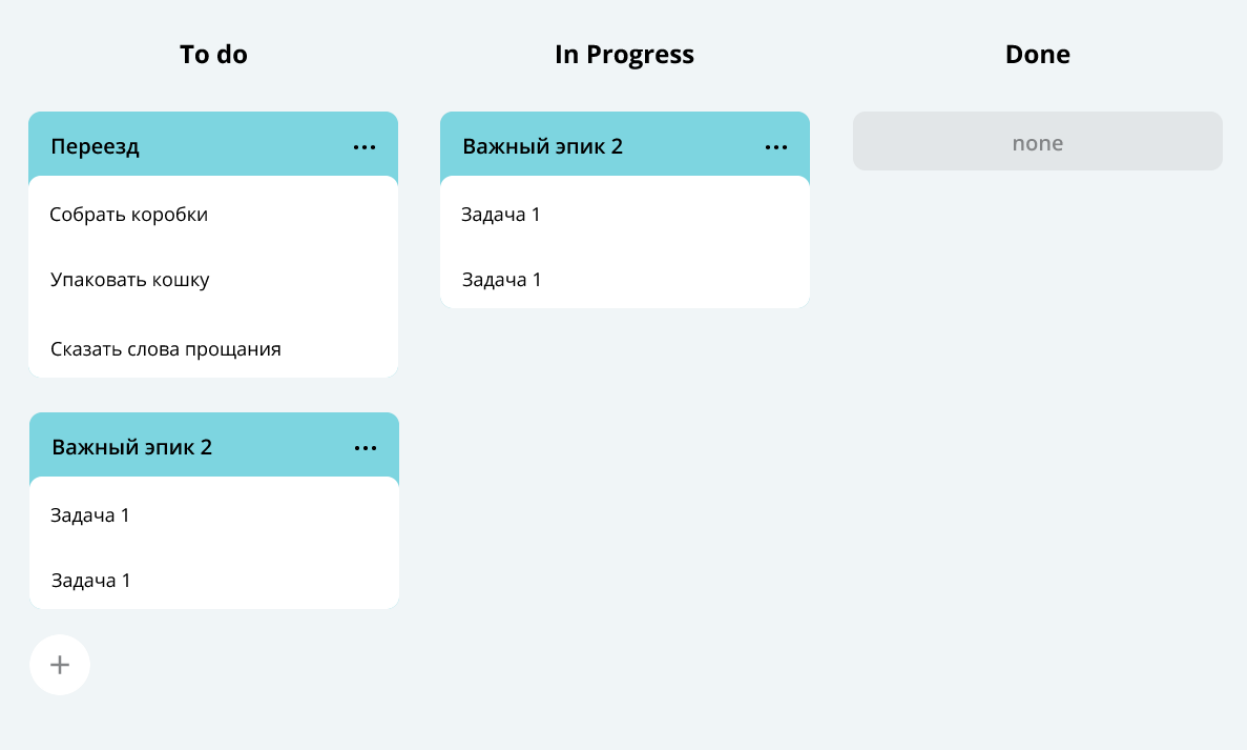
Техническое задание

Как человек обычно делает покупки? Если ему нужен не один продукт, а несколько, то очень вероятно, что сначала он составит список, чтобы ничего не забыть. Сделать это можно где угодно: на листе бумаги, в приложении для заметок или, например, в сообщении самому себе в мессенджере.
А теперь представьте, что это список не продуктов, а полноценных дел. И не каких-нибудь простых вроде «помыть посуду» или «позвонить бабушке», а сложных — например, «организовать большой семейный праздник» или «купить квартиру». Каждая из таких задач может разбиваться на несколько этапов со своими нюансами и сроками. А если над их выполнением будет работать не один человек, а целая команда, то организация процесса станет ещё сложнее.

Трекер задач

Как системы контроля версий помогают команде работать с общим кодом, так и трекеры задач позволяют эффективно организовать совместную работу над задачами. Вам предстоит написать бэкенд для такого трекера. В итоге должна получиться программа, отвечающая за работу этой страницы:



Типы задач

Простейшим кирпичиком такой системы является **задача** (англ. *task*). У задачи есть следующие свойства:

1. **Название**, кратко описывающее суть задачи (например, «Переезд»).
2. **Описание**, в котором раскрываются детали.
3. **Уникальный идентификационный номер задачи**, по которому её можно будет найти.
4. **Статус**, отображающий её прогресс. Мы будем выделять следующие этапы жизни задачи:
    1. `NEW` — задача только создана, но к её выполнению ещё не приступили.
    2. `IN_PROGRESS` — над задачей ведётся работа.
    3. `DONE` — задача выполнена.

Иногда для выполнения какой-нибудь масштабной задачи её лучше разбить на **подзадачи** (англ. *subtask*). Большую задачу, которая делится на подзадачи, мы будем называть **эпиком** (англ. *epic*).
Таким образом, в нашей системе задачи могут быть трёх типов: обычные задачи, эпики и подзадачи. Для них должны выполняться следующие условия:

- Для каждой подзадачи известно, в рамках какого эпика она выполняется.
- Каждый эпик знает, какие подзадачи в него входят.
- Завершение всех подзадач эпика считается завершением эпика.

Менеджер

Кроме классов для описания задач, вам нужно реализовать класс для объекта-менеджера. Он будет запускаться на старте программы и управлять всеми задачами. В нём должны быть реализованы следующие функции:

1. Возможность хранить задачи всех типов. Для этого вам нужно выбрать подходящую коллекцию.
2. Методы:
    1. Получение списка всех задач.
    2. Получение списка всех эпиков.
    3. Получение списка всех подзадач определённого эпика.
    4. Получение задачи любого типа по идентификатору.
    5. Добавление новой задачи, эпика и подзадачи. Сам объект должен передаваться в качестве параметра.
    6. Обновление задачи любого типа по идентификатору. Новая версия объекта передаётся в виде параметра.
    7. Удаление ранее добавленных задач — всех и по идентификатору.
3. Управление статусами осуществляется по следующему правилу:

    1. Менеджер сам не выбирает статус для задачи. Информация о нём приходит менеджеру вместе с информацией о самой задаче.
    2. Для эпиков:

- если у эпика нет подзадач или все они имеют статус `NEW`, то статус должен быть `NEW`.

- если все подзадачи имеют статус `DONE`, то и эпик считается завершённым — со статусом `DONE`.
- во всех остальных случаях статус должен быть `IN_PROGRESS`.

И ещё кое-что...

Не оставляйте в коде мусор — превращённые в комментарии или ненужные куски кода.
Это сквозной проект, на его основе вы будете делать следующие домашние задания.
Давайте коммитам осмысленные комментарии: порядок в репозитории и коде — ключ к успеху написания хороших программ.
Интересного вам программирования!