ЗАДАНИЕ К УРОКУ 2: ПРОЕКТ API ДЛЯ СИСТЕМЫ УЧЕТА РАБОТЫ

* ниже обозначается primary key; он не указыается явно в POST-запросе на создание, поскольку БД присваивает его автоматически, гарантируя уникальность. Далее в других запросах он используется.

Предполагается следующая модель работы, отмечаемая соответствующими статусами контрактов и назначений:

- создаются справочники клиентов и сотрудников, они по необходимости пополняются
- клиент создает контракты
- в рамках контракта между сотрудниками распределяются задачи с указанием их доли в оплате (возможно, оплата должна распределяться и в конце в отношениях затраченного времени)
- после утверждения распределения контракт переводится в состояние "работники назначены" и они приступают к работе
- после выполнения работы назначения переводятся в состояние ожидания оплаты
- после оплаты всех сотрудников назначения и контракт могут быть автоматически удалены из базы

Таблица Clients (справочник по имеющимся клиентам)

* int id
string name

API для работы с таблицей клиентов

GET /api/clients/list/
возвращает список всех клиентов

GET /api/clients/get/{id}
возвращает имя клиента с указанным id

POST /api/clients/add/{name}
добавляет нового клиента

PUT /api/clients/rename/{id}/{name}
переименовывает клиента

DELETE /api/clients/delete/{id}
удаляет клиента; при этом вызывает /api/contract/cleanup/client/ для этого клиента после чего проверяет и отказывает в удалении, если у клиента остались контракты

Таблица Employees (справочник по имеющимся сотрудникам)

* int id
string name

API для работы с таблицей сотрудников

GET /api/employees/list/
возвращает список всех сотрудников

GET /api/employees/get/{id}
возвращает имя сотрудника с указанным id

POST /api/employees/add/{name}
добавляет нового сотрудника

PUT /api/employees/rename/{id}/{name}
переименовывает сотрудника

DELETE /api/employees/delete/{id}
удаляет сотрудника; отказывает в удалении, если у сотрудника остались работы

Таблица Contracts (список контрактов)

* int id
int clientId
string description
int payment
int status

Нетривиальным элементом здесь является статус. Он может принимать только небольшое количество значений, в C# ему соответствует enum:

```
enum ContractStatus {
    ContractCreated,          // Контракт создан, сотрудники пока не назначены
    EmployeesAssigned,        // Задачи распределены между сотрудниками
    ContractCompleted         // Работа выполнена и оплачена, контракт может быть удален
}
```

API для работы с таблицей контрактов

GET /api/contracts/list
возвращает список всех контрактов

GET /api/contracts/status/{status}
возвращает список всех контрактов с указанным статусом

GET /api/contracts/client/{clientId}
возвращает список всех контрактов для указанного клиента

GET /api/contracts/clientstatus/{clientId}/{status}
возвращает список всех контрактов для указанного клиента с указанным статусом

POST /api/contracts/create/{clientId}/{description}/{payment}
создает новый контракт со статусом ContractCreated

PUT /api/contracts/edit/{contractId}/{descriptionNew}/{paymentNew}
изменяет параметры контракта;
их нельзя изменить после перевода контракта в статус EmployeesAssigned

DELETE /api/contracts/delete/{contractId}
удаляет контракт в статусе ContractCreated (со всеми назначениями)
или ContractCompleted;
в статусе EmployeesAssigned пробует сделать /api/contracts/cleanup/
и возвращает ошибку, если не получилось

PUT /api/contracts/status/employedassigned/{contractId}
переводит контракт в статус EmployeesAssigned
все назначения, связанные с контрактом, переводятся в статус AssignmentConfirmed
нельзя сделать, если сумма долей назначенных сотрудников не равна 100%

PUT /api/contracts/cleanup/contract/{contractId}
если остались связанные с контрактом незаверншенные назначения
(не в состоянии PaymentsCompleted), возвращает ошибку
иначе удаляет все связанные с контрактом назначения
и переводит его в статус ContractCompleted

DELETE /api/contracts/cleanup/client/{clientId}
для всех контрактов клиента, кроме находящихся в статусе ContractCreated,
пробует сделать /api/contracts/delete/ (удаляя все завершенные контракты)

DELETE /api/contracts/cleanup/all
для всех контрактов, кроме находящихся в статусе ContractCreated,
пробует сделать /api/contracts/delete/ (удаляя все завершенные контракты)

Таблица Assignments (список назначений сотрудников на контракт)

* int id
int contractId
int employeeId
string taskDescription
int paymentShare
int status

Таблица позволяет назначать сотрудников на контракт, указывая их задачи и доли в общей выплате (в процентах, для простоты целые). Статусу также соответствует enum в C# :

```
enum AssignmentStatus {
    AssignmentCreated,      // Назначение создано
    AssignmentConfirmed,    // Назначение подтверждено
    WorkCompleted,          // Работа выполнена
    PaymentsCompleted       // Работа оплачена
}
```

"Назначение подтверждено" означает, что контракт переведен в статус EmployeesAssigned. После этого менять параметры договора уже нельзя, и сотрудники могут приступать к работе.

API для работы с таблицей назначений

GET /api/assignments/list
возвращает список всех назначений

GET /api/assignments/list/status/{status}
возвращает список всех назначений с указанным статусом

GET /api/assignments/contract/{contract Id}
возвращает список всех назначений для указанного контракта

GET /api/assignments/contractstatus/{contract Id}/{status}
возвращает список всех назначений для указанного контракта с указанным статусом

GET /api/assignments/employee/{employeeId}
возвращает список всех назначений для указанного сотрудника

GET /api/assignments/employeestatus/{employeeId}/{status}
возвращает список всех назначений для указанного сотрудника с указанным статусом
например, сотрудник может посмотреть, какие назначенные ему задачи пока не сделаны
или какие сделанные им работы еще не оплачены

GET /api/assignments/client/{clientId}
возвращает список всех работ для указанного клиента

GET /api/assignments/clientstatus/{clientId}/{status}
возвращает список всех назначений для указанного клиента с указанным статусом
например, какие назначения по всем заказам клиента остались несделанными
в частности, счет на оплату генерируется через этот запрос,
через список всех НЕОПЛАЧЕННЫХ назначений для данного клиента

POST /api/assignments/create/{contractId}/{employeeId}/{description}/{payment}
создает новое назначение со статусом AssignmentCreated

PUT /api/assignments/edit/{assignmentId}/{employeeId}/{description}/{payment}
изменяет параметры назначения;
их нельзя изменить после перевода назначения в статус AssignmentConfirmed

DELETE /api/assignments/delete/{id}
удаляет назначение; нельзя после перевода назначения в статус AssignmentConfirmed

PUT /api/assignments/status/workcompleted/{assignmentId}
переводит назначение в статус WorkCompleted
только из статуса AssignmentConfirmed
PUT /api/assignments/status/paymentscompleted/{assignmentId}
переводит назначение в статус PaymentsCompleted
только из статуса WorkCompleted

ПРИМЕЧАНИЕ.
Отметим, что некоторые вызовы проводят проверку корректности и могут завершиться с ошибкой. Я постарался учитывать, чтобы эти проверки были эффективны — по-моему, в рамках предлагаемой архитектуры каждая делается одним подходящим SELECT-запросом к соответствующей БД.