Задание 2. Тайный Санта

Необходимо реализовать RESTFul API сервис для игры в Тайного Санту.

Типы данных:

Тип данных	**Свойства**
Группа	id – идентификатор группы, натуральное число name – название группы, строка description – описание группы, строка participants – участники группы, список объектов типа “Участник”
Участник	id – идентификатор участника, натуральное число name – имя участника, строка wish – пожелание, строка recipient - подопечный, объект типа “Участник”

Возможные действия:

Метод	**Endpoint**	**Пояснение**	**Описание**
POST	/group	В запросе передается body в формате JSON: { “name”: “string”, “description”: “string”, } В ответе ожидается идентификатор созданной группы, например **1**	Добавление группы с возможностью указания названия (name), описания (description) Описание – **не** обязательный параметр, название – обязательный
GET	/groups	В ответе ожидается полный список групп, **без** указания участников в формате JSON: [{ “id”: number, “name”: “string”, “description”: “string” }, …]	Получение краткой информации о всех группах (**без** информации об участниках)

GET	/group/{id}	ID группы, полную информацию о которой необходимо получить, передается в виде path-параметра, например: **/group/1** В ответе ожидается полная информация о группе, **с** указанием участников в формате JSON: { “id”: number, “name”: “string”, “description”: “string”, “participants”: [{ “id”: number, “name”: “string” “wish”: “string”, “recipient”: { “id”: number, “name”: “string” “wish”: “string” //Здесь не нужно передавать подопечного } }, …] }	Получение полной информации (**с** информацией об участниках) о группе по идентификатору *До проведения жеребьевки *recipient* у участников не заполнен

PUT	/group/{id}	ID редактируемой группы, передается как path-параметр, например: **/group/1** Редактируемые свойства передаются в body запроса в формате JSON: { “name”: “string”, “description”: “string” }	Редактирование группы по идентификатору группы Редактировать можно только свойства name, description Удалить название таким образом нельзя, описание – можно
DELETE	/group/{id}	ID удаляемой группы передается как path-параметр, например: **/group/1**	Удаление группы по идентификатору
POST	/group/{id}/participant	ID группы, в которую добавляется участник, передается как path-параметр: например: **/group/1/participant** В запросе передается body с информацией о добавляемом участнике в формате JSON: { “name”: “string”, “wish”: “string” }	Добавление участника в группу по идентификатору группы Пожелания – **не** обязательный параметр, имя – обязательный

		В ответе ожидается идентификатор добавленного участника, например: **1223**	
DELETE	/group/{groupId}/participant/{participantId}	ID редактируемой группы и ID удаляемого участника передаются как path-параметры, например: **/group/1/participant/1223**	Удаление участника из группы по идентификаторам группы и участника
POST	/group/{id}/toss	ID группы передается в виде path-параметра, например: **/group/1/toss** В ответе ожидается список объектов типа “Участник” с указанными подопечными в формате JSON: [{ “id”: number, “name”: “string” “wish”: “string”, “recipient”: { “id”: number,	Проведение жеребьевки в группе по идентификатору группы Проведение жеребьевки возможно только в том случае, когда количество участников группы >= 3 *Участнику в качестве подопечного нельзя выдать самого себя *Участник не может быть подопечным одновременно у двух и более участников

		“name”: “string” “wish”: “string”, //Здесь не нужно передавать подопечного } } …] В случае, если проведение жеребьевки в данный момент невозможно – следует вернуть код ответа 409 (Conflict)	
GET	/group/{groupId}/participant/{participantId}/recipient	ID группы и участника передаются в виде path-параметров, например: **/group/1/recipient/2** В ответе ожидается объект типа “Участник” – подопечный участника, чей идентификатор передан, в формате JSON: { “id”: number, “name”: “string” “wish”: “string”, }	Получение информации для конкретного участника группы, кому он дарит подарок.

Сервис должен запускаться на порту 8080.

При доступных и адекватных запросах код ответа должен быть равен 200, 201, 202 или 204.

При недоступных или неадекватных запросах сервис должен возвращать соответствующие сообщения об ошибках, код ответа должен отличаться от 200, 201, 202 или 204.

Можно использовать любые open source библиотеки.

Проверка будет производиться автоматизированным тестирующим ПО. Необходимо также приложить исходный код в виде архива и выгрузить его в любой git-репозиторий (github, gitlab, bitbucket) с предоставлением публичного всеобщего доступа и приложить ссылку.

Критерии оценки:

1. Проверка автоматизированным тестирующим ПО.
2. Описание технического решения
   2.1. Соответствие принципам SOLID
   2.2. Описание документации API в одной из общеиспользуемых спецификаций (рекомендуется OpenAPI (Swagger))

Требования к структуре оформления решения:

1. Исходный код решения.
   1.1. Оценка формируется исходя из результатов проверки API автотестом
   1.2. Возможность запуска инфраструктуры приложения в контейнерах (docker-compose)
2. Описание технического решения:
   2.1. Соответствие принципам SOLID
   2.2. Описание документации API в одной из общеиспользуемых спецификаций (рекомендуется OpenAPI (Swagger))

   2.3. Описание настройки среды выполнения и запуска приложения