Хоккейная федерация. API

Автор	**@Mr_Karpets**
Задача	https://github.com/mr-karpets/Portfolio/blob/main/%D0%97%D0%B0%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20-%20%D0%A5%D0%BE%D0%BA%D0%BA%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F%20%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D1%8F/README.md

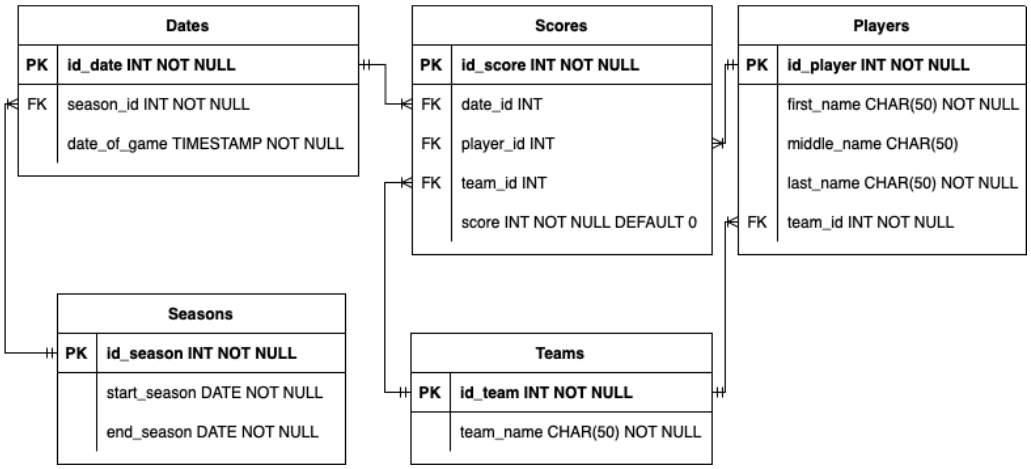
Краткое описание

В рамках задачи необходимо разработать API для системы учета соревнований хоккейной федерации. Запросы должны обеспечивать выполнение следующих функций:

1. Создание новой команды;

2. Добавление нового игрока в команду;
3. Перевод игрока из одной команды в другую;
4. Регистрация новой игры;
5. Фиксация забитой шайбы;
6. Получение счета команды в определенной игре;
7. Получение количества игр игрока за сезон;
8. Получение лучшего бомбардира сезона.

Модель данных



Ограничения

Идентификатор	Ограничение
CONST-01	Модель данных и описанные методы обеспечивают выполнение только тех функций, что перечисленны в кратком описании. Примечание: другие функции и соответствующие методы (например, заявление состава игроков команды на игру и др.) в ТЗ не описаны.
CONST-02	В одно время проходит максимум одна игра (источник - заказчик (организатор соревнования)).
CONST-03	В рамках федерации нет игроков с одновременно совпадающими именем, отчеством и фамилией (источник - заказчик (организатор соревнования)).
CONST-04	Границы сезонов не будут иметь пересечений (источник - заказчик (организатор соревнования)).

Требования к API

Создание новой команды POST/api/hockey_federation/teams/{team_name}

Описание метода

Метод предназначен для регистрации новой команды в базе данных (БД) хоккейной федерации. Каждая команда должна иметь уникальное название, состоящее не более чем из 50 символов. В названии допускается использование цифр, специальных символов, символов латинского алфавита. Успешный вызов метода добавляет строку в таблицу teams.

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	teamName	string(50)	true	Название команды
	Выходные параметры			
2	Meta	object	true	Результат вызова метода
2.1	status	string	true	Код результата
2.2	description	string	true	Пользовательское сообщение

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’.
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность название команды (teamName):
   a. Если название не заполнено или в качестве названия команды указана пустая строка (‘’) или название команды состоит больше чем из 50 символов, то завершать вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное название команды’.
3. Если проверки валидности названия команды успешно пройдены, необходимо выполнить запрос добавления новой команды в БД: INSERT INTO teams (team_name) VALUES (‘{teamName}’), где '{teamName}' - входной параметр teamName из метода:
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, meta.status==‘ERROR', meta.description==’Ошибка сервиса’;
   b. Если команда с таким названием уже зарегистрирована, то завершать вызов метода ошибкой с кодом 409, meta.status==’CONFLICT’, meta.description==’Команда с таким названием уже зарегистрирована’;
   c. Если добавление команды выполнено успешно, то завершать вызов метода успешным ответом с кодом 201, meta.status==’CREATED', meta.description==’Команда зарегистрирована’.

Добавление игрока в команду POST/api/hockey_federation/players/{player}

Описание метода

Метод предназначен для добавления игрока в команду. В федерации не может быть человека с одновременно одинаковыми именем, отчеством и фамилией. Отчество не обязательный атрибут. Команды должны быть предварительно зарегистрированы в справочнике команд федерации. Успешный вызов метода добавляет строку в таблицу players.

№	Название	Тип	Обязательность	Описание

	Входные параметры (body)			
1	firstName	string(50)	true	Имя
2	middleName	string(50)	false	Отчество
3	lastName	string(50)	true	Фамилия
4	currentTeam	string(50)	true	Название команды
	Выходные параметры			
5	Meta	object	true	Результат вызова метода
5.1	status	string	true	Код результата
5.2	description	string	true	Пользовательское сообщение

Логика работы:

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’.
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность имя игрока (firstName):
   a. Если фамилия не заполнена или в качестве фамилии указана пустая строка (‘’) или фамилия состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное имя игрока’;
3. Если проверка на валидность имени игрока пройдена успешно, необходимо проверить на валидность отчество игрока (middleName):
   a. Если отчество состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное отчество игрока’;
4. Если проверка на валидность отчества игрока пройдена успешно, необходимо проверить на валидность фамилию игрока (lastName):
   a. Если фамилия не заполнена или в качестве фамилии указана пустая строка (‘’) или фамилия состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная фамилия игрока’;
5. Если проверки валидности фамилии игрока успешно пройдены, необходимо проверить на валидность название будущей команды игрока (currentTeam):
   a. Если название не заполнено или в качестве названия команды указана пустая строка (‘’) или название команды состоит больше чем из 50 символов, то завершать вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное название будущей команды’;
6. Если проверки валидности названия команды успешно пройдены, необходимо проверить, что указанная команда зарегистрирована в БД федерации:
   SELECT * FROM teams WHERE team_name=’{currentTeam}’,
   где ’{currentTeam}’ - входной параметр currentTeam из метода:
   a. Если команда с указанным названием не зарегистрирована (пустой результат SELECT), то завершить вызов метода ответом с кодом 204, meta.status==’NO CONTENT', meta.description==’Указанная команда не зарегистрирована в базе данных’.
7. Если проверки 1-6 успешно пройдены, необходимо выполнить запрос добавления нового игрока в команду в БД:
   INSERT INTO players (first_name, middle_name, last_name, current_team) VALUES (‘{firstName}', '{middleName}', '{lastName}', (SELECT id_team FROM Teams WHERE team_name='{currentTeam}')), где '{firstName}', '{middleName}', '{lastName}', '{currentTeam}' - входные параметры firstName, middleName, lastName, currentTeam из метода соответственно.
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503,

meta.status==‘ERROR', meta.description==’Ошибка сервиса’;
b. Если запись о регистрации игрока с указанным именем, отчеством и фамилией уже существует, то завершать вызов метода ошибкой с кодом 409, meta.status==’CONFLICT'', meta.description==’Игрок уже зарегистрирован’;
c. Если добавление игрока в команду выполнено успешно, то завершать вызов метода успешным ответом с кодом 201, meta.status==’CREATED'', meta.description==’Игрок зарегистрирован’.

Перевод игрока в другую команду PUT/api/hockey_federation/players/{player}

Описание метода

Метод предназначен для перевода игрока из одной команды в другую. Команда в которую мы переводим игрока, должна быть предварительно зарегистрирована в справочнике команд федерации. Успешный вызов метода изменяет строку в таблицу players.

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	firstName	string(50)	true	Имя
2	middleName	string(50)	false	Отчество
3	lastName	string(50)	true	Фамилия
4	newTeam	string(50)	true	Название команды
	Выходные параметры			
5	Meta	object	true	Результат вызова метода
5.1	status	string	true	Код результата
5.2	description	string	true	Пользовательское сообщение

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’.
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить, что указанный игрок уже зарегистрирован в нашей Федерации:
   SELECT * FROM players WHERE first_name=’{firstName}’ AND middle_name='{middleName}' AND last_name='{lastName}', где '{firstName}', '{middleName}', '{lastName}' - входные параметры firstName, middleName, lastName из метода соответственно.
   a. Если игрок с указанными параметрами не зарегистрирована (пустой результат SELECT), то завершить вызов метода ответом с кодом 204, meta.status==’NO CONTENT', meta.description==’Игрок с данным именем не существует’.
3. Если проверки валидности игрока успешно пройдены, необходимо проверить на валидность название будущей команды игрока (newTeam):
   a. Если название не заполнено или в качестве названия команды указана пустая строка (‘') или название команды состоит больше чем из 50 символов, то завершать вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное название новой команды’;
4. Если проверки валидности названия команды успешно пройдены, необходимо проверить, что указанная команда зарегистрирована в БД федерации:
   SELECT * FROM teams WHERE team_name=’{newTeam}’,
   где ’{newTeam}’ - входной параметр newTeam из метода:

a. Если команда с указанным названием не зарегистрирована (пустой результат SELECT), то завершить вызов метода ответом с кодом 204, meta.status==’NO CONTENT', meta.description==’Текущая команда с указанным названием не зарегистрирована’.

5. Если проверки 1-4 успешно пройдены, необходимо выполнить запрос добавления игрока в новую команду в БД:
   UPDATE players
   SET current_team = (SELECT id_team FROM Teams WHERE team_name=‘{newTeam}’)
   WHERE first_name=’{firstName}’ AND middle_name='{middleName}' AND last_name='{lastName}', где ‘{newTeam}’, '{firstName}', '{middleName}', '{lastName}' - входные параметры newTeam, firstName, middleName, lastName из метода соответственно.
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, meta.status==‘ERROR', meta.description==’Ошибка сервиса’;
   b. Если добавление игрока в команду выполнено успешно, то завершать вызов метода успешным ответом с кодом 200, meta.status==’CREATED'', meta.description==’Игрок зарегистрирован в новой команде’.

Регистрация нового сезона POST/api/hockey_federation/seasons/{season}

Описание метода

Метод предназначен для регистрации даты проведения нового сезона игр в базе данных хоккейной федерации. Границы сезона (начало и конец), не должны пересекаться между собой. Формат дат начала и конца сезона - строка ‘dd.mm.yyyy’. Успешный вызов метода добавляет строку в таблицу seasons.

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	startSeason	string(10)	true	Начало сезона
2	endSeason	string(10)	true	Конец сезона
	Выходные параметры			
3	Meta	object	true	Результат вызова метода
3.1	status	string	true	Код результата
3.2	description	string	true	Пользовательское сообщение

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’.
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить валидность указанного начало сезона:
   a. Если дата конца сезона не заполнено или в качестве даты указана пустая строка (‘’) или дата не соответствует формату ‘dd.mm.yyyy’, то завершить вызов

   метода ошибкой с кодом 400, meta.status==’ERROR',

   meta.description==’Некорректная дата начала сезона’;
   b. Если указанная дата начала сезона находиться внутри границ другого сезона, т.е. результат запроса будет больше нуля:
   (SELECT SUM(IF('{startSeason}' BETWEEN start_season AND end_season, 1, 0)) FROM Seasons) > 0,
   где '{startSeason}' входной параметр startSeason из Метода.
   То, завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата начала сезона’;
3. Если проверки валидности начала сезона успешно пройдены, необходимо проверить валидность указанного конца сезона:
   a. Если дата конца сезона не заполнено или в качестве даты указана пустая строка (‘’) или дата не соответствует формату

‘dd.mm.yyyy’, то завершить вызов

метода ошибкой с кодом 400, meta.status==’ERROR',

meta.description==’Некорректная дата конца сезона’;
b. Если указанная дата конца сезона находиться внутри границ другого сезона, т.е. результат запроса будет больше нуля:
(SELECT SUM(IF('{endSeason}' BETWEEN start_season AND end_season, 1, 0)) FROM Seasons) > 0,
где '{endSeason}' входной параметр endSeason из Метода.
То, завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата конца сезона’;

4. Если проверки 1-3 пройдены успешно, необходимо выполнить запрос добавления нового сезона в БД:
INSERT INTO Seasons (start_season, end_season) VALUES ('{startSeason}', '{endSeason}'), где '{startSeason}', '{endSeason}' - входные параметры startSeason, endSeason из метода соответственно:
a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, ‘ERROR',
meta.description==’Ошибка сервиса’;
b. Если добавление игры выполнено успешно, то завершать вызов метода успешным ответом с кодом 201,
meta.status==’CREATED'', meta.description==’Игра зарегистрирована’;

Регистрация новой игры POST/api/hockey_federation/Dates/{date_of_game}

Описание метода

Метод предназначен для регистрации даты и времени проведения новой игры в базе данных хоккейной федерации. Игра должна находиться во временных границах уже существующего в базе данных сезона. Формат времени проведения новой игры - строка ‘dd.mm.yyyy hh:mm’. Успешный вызов метода добавляет строку в таблицу Dates.

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	dateOfGame	string(16)	true	Время и дата игры
	Выходные параметры			
2	Meta	object	true	Результат вызова метода
2.1	status	string	true	Код результата
2.2	description	string	true	Пользовательское сообщение

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED',
   meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’;
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность дату игры (dateOfGame):
   a. Если дата не заполнена или в качестве даты указана пустая строка (‘’) или дата не соответствует формату ‘dd.mm.yyyy hh:mm’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата игры’;
   b. Если указанная дата не находиться внутри границ сезона, т.е. запрос не даст результата:
   SELECT id_season FROM Seasons WHERE ‘{dateOfGame}' BETWEEN start_season AND end_season,
   где '{dateOfGame}' входной параметр dateOfGame из Метода.
   То, завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата игры или границы сезона’;
   c. Если игра с указанным временем начала игры уже зарегистрирована
   (т.е. запрос не пустой: SELECT * FROM Dates WHERE date_of_game = '{dateOfGame}’, где '{dateOfGame}' входной параметр

dateOfGame из Метода),
то завершать вызов метода ошибкой с кодом 409, meta.status==’CONFLICT'', meta.description==’Игра с указанной датой уже зарегистрирована’;

3. Если проверки 1-2 пройдены успешно, необходимо выполнить запрос добавления новой игры в БД: INSERT INTO Dates (season_id, date_of_game) VALUES ((SELECT id_season FROM Seasons WHERE ‘{dateOfGame}' BETWEEN start_season AND end_season), '{dateOfGames}'), где '{dateOfGames}' - входные параметры dateOfGames из метода соответственно:
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, ‘ERROR', meta.description==’Ошибка сервиса’;
   b. Если добавление игры выполнено успешно, то завершать вызов метода успешным ответом с кодом 201, meta.status==’CREATED'', meta.description==’Игра зарегистрирована’;

Фиксация забитой шайбы POST/api/hockey_federation/scores/{score}

Описание метода

Метод предназначен для фиксации забитых шайб определенным игроком определенной команды в определенной игре. Название команды, фамилия игрока, даты игры должны быть валидны и предварительно зарегистрированы в БД федерации.

Предусловие: на момент начала игры для каждого заявленного на конкретную игру игрока в БД существует запись в таблице Scores с score=0 (CONST-01)

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	dateOfGame	string(16)	true	Время и дата игры
2	team	string(50)	true	Название команды
3	firstName	string(50)	true	Имя игрока
4	middleName	string(50)	false	Отчество игрока
5	lastName	string(50)	true	Фамилия игрока
6	newScore	integer	true	Количество забитых шайб
	Выходные параметры			
7	Meta	object	true	Результат вызова метода
7.1	status	string	true	Код результата
7.2	description	string	true	Пользовательское сообщение

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’;
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность дату игры (dateOfGame):
   a. Если дата не заполнена или в качестве даты указана пустая строка (‘') или дата не соответствует формату ‘dd.mm.yyyy hh:mm’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата игры’;
3. Если проверки валидности даты игры успешно пройдены, необходимо проверить, что указанная дата игры зарегистрирована в БД федерации: SELECT * FROM games WHERE date_of_game=’{dateOfGame}’, где ’{dateOfGame}’ - входной параметр

dateOfGame из метода соответственно:
a. Если игра с указанной датой не зарегистрирована (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игра с указанной датой не зарегистрирована’;

4. Если проверки валидности и корректности даты игры успешно пройдены, необходимо проверить на валидность название команды игрока (team):
   a. Если название не заполнено или в качестве названия команды указана пустая строка (‘’) или название команды состоит больше чем из 50 символов, то завершать вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное название команды’;
5. Если проверки валидности названия команды успешно пройдены, необходимо проверить, что указанная команда зарегистрирована в БД федерации: SELECT * FROM teams WHERE name=’{team}’, где ’{team}’ - входной параметр team из метода соответственно:
   a. Если команда с указанным названием не зарегистрирована (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Команда с указанным названием не зарегистрирована’;
6. Если проверки валидности и корректности названия команды успешно пройдены, необходимо проверить на валидность параметр newScore:
   a. Если параметр не заполнен или указана пустая строка ('') или параметр больше 2 символов или содержит иные символы кроме чисел, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректно введён счёт игрока’;
7. Если проверки валидности и корректности параметра newScore успешно пройдены, необходимо проверить на валидность имя игрока (firstName):
   a. Если фамилия не заполнена или в качестве фамилии указана пустая строка (‘’) или фамилия состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное имя игрока’;
8. Если проверка на валидность имени игрока пройдена успешно, необходимо проверить на валидность отчество игрока (middleName):
   a. Если отчество состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное отчество игрока’;
9. Если проверка на валидность отчества игрока пройдена успешно, необходимо проверить на валидность фамилию игрока (lastName):
   a. Если фамилия не заполнена или в качестве фамилии указана пустая строка (‘’) или фамилия состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная фамилия игрока’;
10. Если проверки валидности игрока успешно пройдены, необходимо проверить, что игрок зарегистрирован базе данных и в указанной команде: (
    SELECT * FROM players WHERE first_name=’{firstName}’ AND middle_name=’{middleName}’ AND last_name=’{lastName}’
) = ’{team}’, где '{firstName}', '{middleName}', '{lastName}', '{team}' - входные параметры firstName, middleName, lastName, team из метода соответственно:
    a. Если указанный игрок в базе не числиться (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игрок не зарегистрирован в базе данных’;
    b. Если указанный игрок в текущей команде не зарегистрирован (запрос SELECT не равен team), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игрок не зарегистрирован в указанной команде’;
11. Если проверки 1-7 успешно пройдены, необходимо выполнить запрос проверки регистрации игрока на данную игру:
    SELECT * FROM Scores WHERE date_of_game='{dateOfGame}'
    AND team_id=(SELECT id_team FROM Teams WHERE team_name='{team}')
    AND player_id=(SELECT id_player FROM players WHERE first_name='{firstName}' AND middle_name='{middleName}' AND last_name='{lastName}')
    , где ‘{dateOfGame}, ‘{team}’, ‘{firstName}, '{lastName}’, '{middleName}’ - входные параметры dateOfGame, team, firstName, lastName, middleName из метода соответственно:

a. Если указанный игрок не заявлен на игру (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игрок не был заявлен в составе команды для указанной игры’;

12. Если проверки 1-8 успешно пройдены, необходимо выполнить запрос фиксации новой шайбы в БД: UPDATE Scores SET score='{newScore}' WHERE date_of_game='{dateOfGame}'
    AND team_id=(SELECT id_team FROM Teams WHERE team_name='{team}')
    AND player_id=(SELECT id_player FROM players WHERE first_name='{firstName}' AND middle_name='{middleName}' AND last_name='{lastName}'), где ‘{newScore}', ‘{dateOfGame}, ‘{team}', ‘{firstName}, ‘{lastName}', ‘{middleName}’ - входные параметры dateOfGame, team, firstName, lastName, middleName из метода соответственно:
13. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, ‘ERROR', meta.description==’Ошибка сервиса’;
14. Если фиксация шайбы выполнена успешно, то завершать вызов метода успешным ответом с кодом 200, meta.status==’OK', meta.description==’Счёт зафиксирован’;

Получение счета команды в определенной игре GET/api/hockey_federation/scores/sum

Описание метода

Метод предназначен для получения суммы забитых голов игроков определенной команды в определенной игре. Название команды и даты игры должны быть валидны и предварительно зарегистрированы в БД федерации.

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	dateOfGame	string(16)	true	Время и дата игры
2	team	string(50)	true	Название команды
	Выходные параметры			
3	Meta	object	true	Результат вызова метода
3.1	status	string	true	Код результата
3.2	description	string	true	Пользовательское сообщение
4	Data	object	false	Статистические данные
4.1	score	string	false	Счёт команды в игре

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’;
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность дату игры (dateOfGame):
   a. Если дата не заполнена или в качестве даты указана пустая строка (‘') или дата не соответствует формату ‘dd.mm.yyyy hh:mm’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата игры’;
3. Если проверки валидности даты игры успешно пройдены, необходимо проверить, что указанная дата игры зарегистрирована в БД федерации: SELECT * FROM Dates WHERE date_of_game=’{dateOfGame}’, где ’{dateOfGame}’ - входной параметр dateOfGame из метода:
   a. Если игра с указанной датой не зарегистрирована (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игра с указанной датой не зарегистрирована’;

4. Если проверки валидности и корректности даты игры успешно пройдены, необходимо проверить на валидность название команды (name):
   a. Если название не заполнено или в качестве названия команды указана пустая строка (‘’) или название команды состоит больше чем из 50 символов, то завершать вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное название команды’;
5. Если проверки валидности названия команды успешно пройдены, необходимо проверить, что указанная команда зарегистрирована в БД федерации: SELECT * FROM teams WHERE name=’{team}’, где ’{team}’ - входной параметр team из метода соответственно:
   a. Если команда с указанным названием не зарегистрирована (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Команда с указанным названием не зарегистрирована’;
6. Если проверки 1-5 успешно пройдены, необходимо выполнить запрос получения счета команды в игре:
   SELECT SUM(score) AS result FROM Scores WHERE date_of_game='{dateOfGame}' AND team_name='{team}',
   где '{dateOfGame}, '{team}' - входные параметры dateOfGame, team из метода соответственно:
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, ‘ERROR', meta.description==’Ошибка сервиса’;
   b. Если команда не заявлена на указанную игру (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Команда с указанным названием не заявлена на данную игру’;
   c. Если счет успешно получен, то завершать вызов метода успешным ответом с кодом 200, meta.status==’OK', meta.description==’Счет:’, data.score==result;

Получение количества игр игрока за сезон GET/api/hockey_federation/view_players/count

Описание метода

Метод предназначен для получения количества игр, на которые был заявлен игрок в пределах сезона в определенной команде. Параметры игрока, границы сезона должны быть валидны и предварительно зарегистрированы в БД федерации.

Для получения необходимых данных мы будем использовать представление view_players:

```
CREATE VIEW view_players AS (
    SELECT start_season, end_season, id_players AS id, first_name, middle_name, last_name, SUM(score) AS score,
COUNT(Scores.date_id) AS Games
    FROM Players
    LEFT JOIN Scores
    ON Players.id_player = Scores.player_id
    LEFT JOIN dates
    ON Scores.date_id = Dates.id_date
    GROUP BY start_season, id
);
```

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	startSeason	string(10)	true	Начало сезона
2	endSeason	string(10)	true	Конец сезона
3	firstName	string(50)	true	Имя игрока
4	middleName	string(50)	false	Отчество игрока
5	lastName	string(50)	true	Фамилия игрока
	Выходные параметры			

6	Meta	object	true	Результат вызова метода
6.1	status	string	true	Код результата
6.2	description	string	true	Пользовательское сообщение
7	Data	object	false	Статистические данные
7.1	count	integer	false	Количество игр игрока

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’;
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность дату начала сезона (startSeason):
   a. Если дата не заполнена или в качестве даты указана пустая строка (‘’) или дата не соответствует формату ‘dd.mm.yyyy’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата начала сезона;
3. Если проверка на валидность начала сезона пройдена, необходимо проверить на валидность дату окончания сезона (endSeason):
   a. Если дата не заполнена или в качестве даты указана пустая строка (‘’) или дата не соответствует формату ‘dd.mm.yyyy’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата окончания сезона’;
4. Если проверки валидности границ сезона успешно пройдены, необходимо проверить на валидность имя игрока (firstName):
   a. Если фамилия не заполнена или в качестве фамилии указана пустая строка (‘’) или фамилия состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное имя игрока’;
5. Если проверка на валидность имени игрока пройдена успешно, необходимо проверить на валидность отчество игрока (middleName):
   a. Если отчество состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректное отчество игрока’;
6. Если проверка на валидность отчества игрока пройдена успешно, необходимо проверить на валидность фамилию игрока (lastName):
   a. Если фамилия не заполнена или в качестве фамилии указана пустая строка (‘’) или фамилия состоит больше чем из 50 символов или фамилия содержит цифры/специальные символы/символы латинского алфавита, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная фамилия игрока’;
7. Если проверки валидности игрока успешно пройдены, необходимо проверить, что игрок зарегистрирован базе данных:
   SELECT * FROM players WHERE first_name=’{firstName}’ AND middle_name=’{middleName}’ AND last_name=’{lastName}’
   , где '{firstName}', '{middleName}', '{lastName}', - входные параметры firstName, middleName, lastName, из метода соответственно:
   a. Если указанный игрок в базе не числиться (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игрок не зарегистрирован в базе данных’;
8. Если проверки 1-7 успешно пройдены, необходимо выполнить запрос получения количества игр, на которые был заявлен указанный игрок в указанном сезоне:
   SELECT Games AS result FROM view_players WHERE start_season='{startSeason}' AND end_season='{endSeason}' AND first_name=’{firstName}’ AND middle_name=’{middleName}’ AND last_name=’{lastName}’
   , где ‘{startSeason}’, ‘{endSeason}’, '{firstName}', '{middleName}', '{lastName}' - входные параметры startSeason, endSeason, firstName, middleName, lastName, из метода соответственно:
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, meta.status==‘ERROR', meta.description==’Ошибка сервиса’;
   b. Если игрок не был заявлен ни в одной игре в течение указанного сезона (пустой результат SELECT), то завершить вызов

метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’Игрок не был заявлен в составе команды в этом сезоне’;

c. Если количество успешно получено, то завершать вызов метода успешным ответом с кодом 200, meta.status==’OK', meta.description==’Количество сыгранных за сезон игр:’, data.count==result;

Получение лучшего бомбардира сезона GET/api/hockey_federation/view_players/{player}

Описание метода

Метод предназначен для получения имени и фамилии лучшего бомбардира сезона. Границы сезона должны быть валидны. Для получения необходимых данных мы будем использовать представление view_players:

```
CREATE VIEW view_players AS (
    SELECT start_season, end_season, id_players AS id, first_name, middle_name, last_name, SUM(score) AS score,
COUNT(Scores.date_id) AS Games
    FROM Players
    LEFT JOIN Scores
    ON Players.id_player = Scores.player_id
    LEFT JOIN dates
    ON Scores.date_id = Dates.id_date
    GROUP BY start_season, id
);
```

№	Название	Тип	Обязательность	Описание
	Входные параметры (body)			
1	startSeason	string(10)	true	Начало сезона
2	endSeason	string(10)	true	Конец сезона
	Выходные параметры			
3	Meta	object	true	Результат вызова метода
3.1	status	string	true	Код результата
3.2	description	string	true	Пользовательское сообщение
4	Data	object	false	Статистические данные
4.1	count	string	false	Имя и фамилия игрока

Логика работы

1. После вызова метода необходимо проверять авторизационный токен пользователя:
   a. Если получен невалидный токен авторизации, то завершать вызов метода ошибкой с кодом 401, meta.status==’FAILED', meta.description==’Ошибка авторизации’;
   b. Если у авторизованного пользователя нет прав на добавление новой команды, то завершать вызов метода ошибкой с кодом 403, meta.status ==’FORBIDDEN’, meta.description==’Недостаточно прав для выполнения операции’;
2. Если проверки авторизационного токена успешно пройдены, необходимо проверить на валидность дату начала сезона (startSeason):
   a. Если дата не заполнена или в качестве даты указана пустая строка (‘’) или дата не соответствует формату ‘dd.mm.yyyy’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата начала сезона;
3. Если проверка на валидность начала сезона пройдена, необходимо проверить на валидность дату окончания сезона (endSeason):

a. Если дата не заполнена или в качестве даты указана пустая строка (‘’) или дата не соответствует формату ‘dd.mm.yyyy’, то завершить вызов метода ошибкой с кодом 400, meta.status==’ERROR', meta.description==’Некорректная дата окончания сезона’;

4. Если проверки валидности дат пройдены, необходимо проверить, что в рамках указанного сезона были зарегистрированы игры:
   SELECT * FROM games WHERE start_season=’{startSeason}’ AND end_season=’{endSeason}’, где ’{startSeason}’, ‘{endSeason}’ входные параметры startSeason, endSeason из метода соответственно:
   a. Если в указанный сезон не было зарегистрировано игр (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’В рамках сезона не было зарегистрировано ни одной игры’;
5. Если проверки 1-4 успешно пройдены, необходимо выполнить запрос получения максимальной суммы голов, забитых одним игроком в рамках одного сезона:
   SELECT CONCAT(first_name, last_name) AS reault FROM view_players
   WHERE score = (SELECT MAX(score) FROM view_players WHERE start_season=’{startSeason}’ AND end_season=’{endSeason}’)
   AND start_season=’{startSeason}’ AND end_season=’{endSeason}’,
   где ’{startSeason}’, '{endSeason}’ входные параметры startSeason, endSeason из метода соответственно:
   a. Если БД не доступна или запрос не ответил по таймауту, то завершать вызов метода ошибкой с кодом 503, ‘ERROR', meta.description==’Ошибка сервиса’;
   b. Если в рамках сезона не было сыграно ни одной игры (пустой результат SELECT), то завершить вызов метода ошибкой с кодом 204, meta.status==’NO CONTENT', meta.description==’В рамках сезона не было сыграно ни одной игры’;
   c. Если максимальная сумма голов успешно получена, то завершать вызов метода успешным ответом с кодом 200, meta.status==’OK', meta.description==’Лучший бомбардир сезона:’, data.score==result;