API документация сервера

Аутентификация

POST /api/auth/register

Описание: Регистрация нового пользователя.
Тело запроса (JSON):

- username: строка, обязательный
- email: строка, обязательный
- password: строка, обязательный
- company_name: строка, опционально

Пример запроса:
POST /api/auth/register

Content-Type: application/json

{

"username": "ivan123",

"email": "ivan@mail.com",

"password": "supersecret",

"company_name": "ООО Ромашка"

}
Пример ответа при успехе:
HTTP/1.1 201 Created

Content-Type: application/json

{

"message": "User registered successfully"

}
Пример ошибки, если такой пользователь уже есть:
HTTP/1.1 409 Conflict

Content-Type: application/json

{

"message": "User with this username or email already exists"

}

POST /api/auth/login

Описание: Вход пользователя.
Тело запроса (JSON):

- username: строка, обязательный
- password: строка, обязательный

Пример запроса:
POST /api/auth/login

Content-Type: application/json

{

"username": "ivan123",

"password": "supersecret"

}
Пример успешного ответа:
HTTP/1.1 200 OK

Content-Type: application/json

{

"token": "<JWT-токен>"

}
Пример ошибки:
HTTP/1.1 401 Unauthorized

Content-Type: application/json

{

"message": "Invalid username or password"

}

Пользователи

Требуется заголовок: Authorization: Bearer <JWT-токен>

GET /api/users

Описание: Получить список всех пользователей (только для залогиненных).
Пример ответа:
HTTP/1.1 200 OK

Content-Type: application/json


```
[
 {
   "id": 1,
   "username": "ivan123",
   "email": "ivan@mail.com",
   "is_admin": false,
   "company_name": "ООО Ромашка",
   "created_at": "2024-06-08T12:00:00.000Z"
 },
 {
   ...
 }
]
```

GET /api/users/:id

Описание: Получить пользователя по ID.
Пример ответа:


```
{
 "id": 1,
 "username": "ivan123",
 "email": "ivan@mail.com",
 "is_admin": false,
 "company_name": "ООО Ромашка",
 "created_at": "2024-06-08T12:00:00.000Z"
}
```


Ошибка, если не найден:

{

"message": "User not found"

}

PUT /api/users/:id

Описание: Изменить email или компанию.
Тело запроса:

- email — строка, опционально
- company_name — строка, опционально

Пример:
PUT /api/users/1

Content-Type: application/json

{

"email": "new.email@site.ru"

}
Ответ:
{

"id": 1,

"username": "ivan123",

"email": "new.email@site.ru",

"is_admin": false,

"company_name": "ООО Ромашка",

"created_at": "2024-06-08T12:00:00.000Z"

}

DELETE /api/users/:id

Описание: Удалить пользователя по ID.
Пример ответа:
{

"message": "User deleted"

}

Команды (Teams)

Все методы требуют авторизации

GET /api/teams

Выдает список всех команд.
Ответ:
[

{

"id": 1,

"developers": [...],

"forms": [...]

}, ...

]

POST /api/teams

Создать новую команду (только админ). Тело — пустое. Возвращается созданная команда:
{

"id": 5,

"developers": [],

"forms": []

}

DELETE /api/teams/:id

Удалить команду.
Ответ:
{

"message": "Team deleted"

}

Разработчики (Developers)

GET /api/developers

Список всех разработчиков (и их команды).
Ответ:
[

```
 {
   "id": 1,
   "name": "Петя",
   "role": "Backend",
   "level": "Senior",
   "hourly_rate_rub": 1200,
   "skills": ["NodeJS", "Postgres"],
   "team": { ... }
 }, ...
]
```

POST /api/developers

Создать разработчика.
Тело:

- name — строка
- role — строка
- level — строка
- hourly_rate_rub — число
- skills — массив строк
- team_id — число (опционально)

Пример:


```
{
 "name": "Вася",
 "role": "Frontend",
 "level": "Middle",
 "hourly_rate_rub": 900,
 "skills": ["React", "CSS"],
 "team_id": 3
}
```


Ответ:


```
{
 "id": 7,
 "name": "Вася",
 "role": "Frontend",
```

...

}

PUT /api/developers/:id

Изменить разработчика.

DELETE /api/developers/:id

Удалить.

Формы (Forms)

GET /api/forms

Выдаёт все формы пользователя (клиента).

POST /api/forms

Тело:

- title — строка
- team_id — число
- deadline — строка (ISO дата)
- total_cost — число
- tasks — массив задач (см. ниже)

Каждая задача:

- name — строка
- developer_id — число (опционально)
- hours — число
- deadline — дата

Пример:

{

"title": "CRM для магазина",

"team_id": 2,

"deadline": "2024-07-01T00:00:00Z",

"total_cost": 560000,

"tasks": [

{

"name": "Разработка фронта",

"developer_id": 4,

"hours": 80,

```
    "deadline": "2024-06-20T15:00:00Z"
  }
 ]
}
```

Ответ:

```
{
 "id": 5,
 "title": "CRM для магазина",
 "team": { ... },
 "deadline": "2024-07-01T00:00:00Z",
 "total_cost": 560000,
 "created_at": "2024-06-20T10:42:00Z",
 "tasks": [...],
 "user": { ... }
}
```

DELETE /api/forms/:id

Удалить форму.

Задачи (Tasks)

GET /api/tasks

Возвращает:

```
[
 {
  "id": 3,
  "developer": { ... },
  "form": { ... },
  "name": "Бэкенд",
  "hours": 40,
  "deadline": "2024-06-13T10:01:00Z",
```

"is_completed": false

},

...

]

POST /api/tasks

Создать задачу.
Тело:

- form_id — число
- developer_id — число (опционально)
- name — строка
- hours — число
- deadline — строка (дата)
- is_completed — boolean (опционально)

Пример:

```
{

 "form_id": 2,

 "developer_id": 1,

 "name": "Анализ требований",

 "hours": 8,

 "deadline": "2024-06-24T18:00:00Z"

}
```

Ответ:

```
{

 "id": 25,

 "developer": {...},

 "form": {...},

 "name": "Анализ требований",

 ...

}
```

PUT /api/tasks/:id

Обновить задачу.

DELETE /api/tasks/:id

Удалить.

Авторизация

Почти все методы (кроме регистрации и логина) требуют авторизации Добавляйте в заголовок:
Authorization: Bearer <ваш JWT токен>

Пример последовательной работы

- Регистрация или вход пользователя (/api/auth/register, /api/auth/login)
- Сохраняете токен из ответа
- К запросам, связанным с вашими сущностями (/api/users, /api/forms ...) добавляете Authorization: Bearer <token>