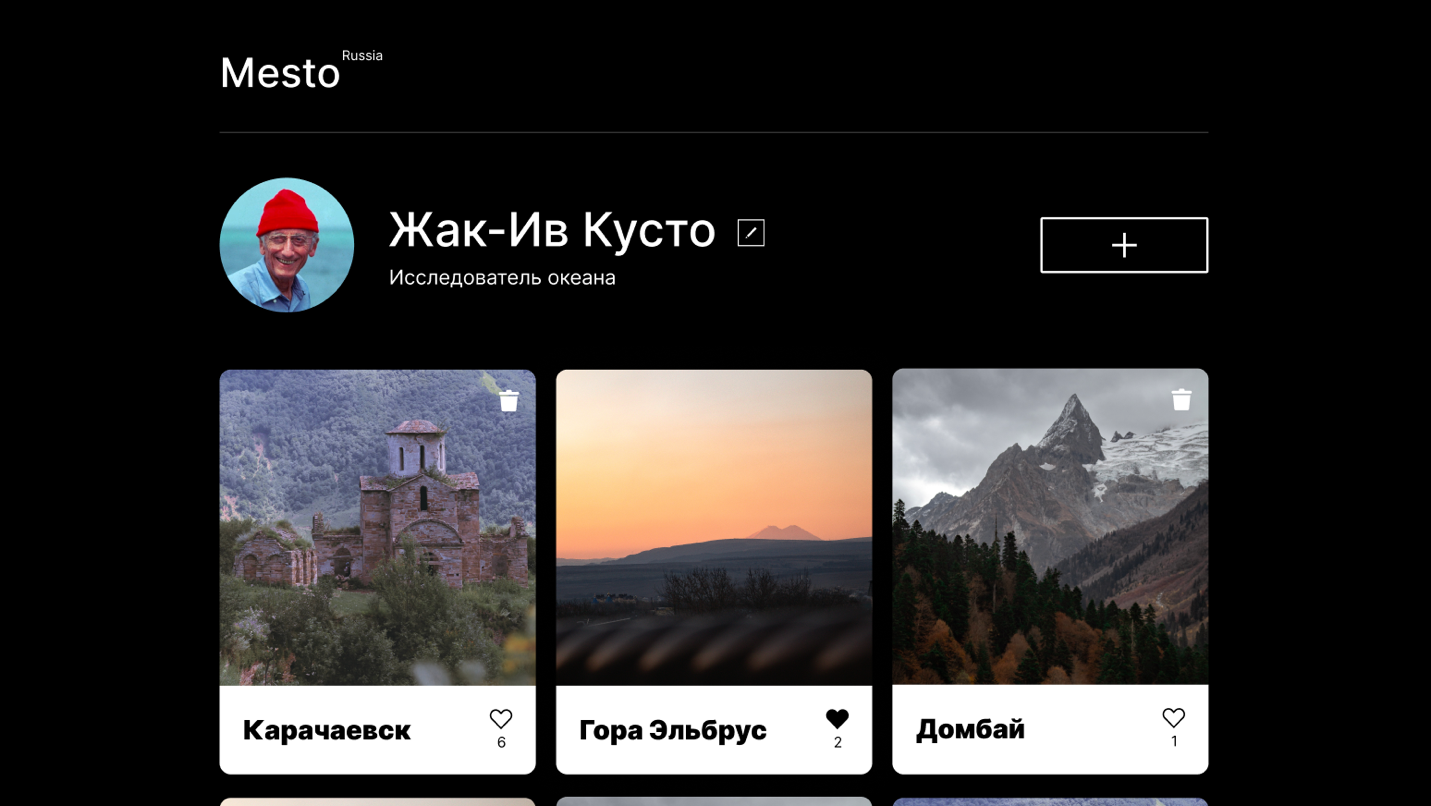
**Проектная работа 9**

В проектной работе этого спринта вы подключите проект Mesto к серверу.



*В этом спринте проект оживёт*

**Необходимая информация**

Для выполнения задания этого спринта вам понадобится дополнительная информация — ваш личный токен и идентификатор вашей группы:

Скопировать код

c56e30dc-2883-4270-a59e-b2f7bae969c6 — так выглядит токен

cohort-42 — а так идентификатор группы

Вы получите их в личных сообщениях в Slack. Если по какой-то причине вы не получили этих данных, напишите куратору.

Токен нужен для того, чтобы сервер умел различать пользователей. Он принадлежит только вам, не делитесь им с другими студентами.

При авторизации на каком-нибудь сайте, например, в социальной сети, обычно вводят логин и пароль. Эти логин и пароль отправляются на сервер, а сервер взамен присылает уникальный токен. Когда вы делаете к серверу новые запросы, они должны содержать этот токен, чтобы сервер понял, что запросы делаете именно вы.

Вы получили этот токен сразу. Мы пропустили шаг с обменом логина и пароля на токен. Иначе всю неделю пришлось бы потратить только на его реализацию.

**Как сделать запрос к серверу**

Адрес сервера проекта Mesto: [https://mesto.nomoreparties.co](https://mesto.nomoreparties.co/).

При каждом запросе нужно передавать токен и идентификатор группы. Вот пример запроса одного студента из группы с идентификатором cohort-42:

Скопировать кодJAVASCRIPT

fetch('https://mesto.nomoreparties.co/v1/cohort-42/cards', {

headers: {

authorization: 'c56e30dc-2883-4270-a59e-b2f7bae969c6'

}

})

.then(res => res.json())

.then((result) => {

console.log(result);

});

Обратите внимание, что при запросе нужно передавать токен в заголовке authorization. Это необходимое условие. Если не передать серверу токен или передать неправильный — сервер ответит ошибкой. Делайте это в каждом запросе.

Идентификатор группы должен быть в URL сразу после v1. Всегда обращайтесь к своей группе. Если обратиться к чужой, сервер вернёт ошибку.

**1. Загрузка информации о пользователе с сервера**

Информация о пользователе должна подгружаться с сервера. Чтобы осуществить это, сделайте GET-запрос на URL (cohortId замените на идентификатор вашей группы):

Скопировать код

GET https://nomoreparties.co/v1/cohortId/users/me

При запросе не забудьте передать токен. Если запрос прошёл успешно, в ответе вы получите объект пользователя:

Скопировать кодJSON

{

"name": "Jacques Cousteau",

"about": "Sailor, researcher",

"avatar": "https://pictures.s3.yandex.net/frontend-developer/ava.jpg",

"\_id": "e20537ed11237f86bbb20ccb",

"cohort": "cohort0"

}

Используйте свойства name, about и avatar в соответствующих элементах шапки страницы. Свойство \_id — это идентификатор пользователя, в данном случае вашего.

**2. Загрузка карточек с сервера**

Начальные карточки должны подгружаться с сервера. Для этого сделайте GET-запрос:

Скопировать код

GET https://mesto.nomoreparties.co/v1/cohortId/cards

В ответ придёт JSON с массивом карточек, которые загрузили студенты вашей группы:

Скопировать кодJSON

[

{

"likes": [],

"\_id": "5d1f0611d321eb4bdcd707dd",

"name": "Байкал",

"link": "https://pictures.s3.yandex.net/frontend-developer/cards-compressed/baikal.jpg",

"owner": {

"name": "Jacques Cousteau",

"about": "Sailor, researcher",

"avatar": "https://pictures.s3.yandex.net/frontend-developer/ava.jpg",

"\_id": "ef5f7423f7f5e22bef4ad607",

"cohort": "local"

},

"createdAt": "2019-07-05T08:10:57.741Z"

},

{

"likes": [],

"\_id": "5d1f064ed321eb4bdcd707de",

"name": "Архыз",

"link": "https://pictures.s3.yandex.net/frontend-developer/cards-compressed/arkhyz.jpg",

"owner": {

"name": "Jacques Cousteau",

"about": "Sailor, researcher",

"avatar": "https://pictures.s3.yandex.net/frontend-developer/ava.jpg",

"\_id": "ef5f7423f7f5e22bef4ad607",

"cohort": "local"

},

"createdAt": "2019-07-05T08:11:58.324Z"

}

]

Используйте этот массив при отображении предзагруженных карточек, а от предыдущего способа отображения первоначальных карточек избавьтесь.

У каждой карточки есть свойства name и link — это заголовок и ссылка на картинку — они понадобятся при отображении каждой отдельной карточки.

Как видите, у карточки также есть идентификатор — свойство \_id. Сейчас он вам не нужен, но скоро понадобится.

**3. Редактирование профиля**

Отредактированные данные профиля должны сохраняться на сервере. Для этого отправьте запрос методом PATCH:

Скопировать код

PATCH https://mesto.nomoreparties.co/v1/cohortId/users/me

В заголовках запроса, кроме токена, необходимо отправить Content-Type, а в теле — JSON с двумя свойствами — name и about. Значениями этих свойств должны быть обновлённые данные пользователя. Вот пример такого запроса:

Скопировать кодJAVASCRIPT

fetch('https://mesto.nomoreparties.co/v1/cohortId/users/me', {

method: 'PATCH',

headers: {

authorization: 'c56e30dc-2883-4270-a59e-b2f7bae969c6',

'Content-Type': 'application/json'

},

body: JSON.stringify({

name: 'Marie Skłodowska Curie',

about: 'Physicist and Chemist'

})

});

Если обновление прошло успешно, в теле ответа от сервера вы получите обновлённые данные пользователя:

Скопировать кодJSON

{

"name": "Marie Skłodowska Curie",

"about": "Physicist and Chemist",

"avatar": "https://pictures.s3.yandex.net/frontend-developer/common/ava.jpg",

"\_id": "e20537ed11237f86bbb20ccb",

"cohort": "cohort0",

}

Метод PATCH обычно используют для обновления сущностей, уже существующих на сервере. Обновление информации о пользователе именно такой случай: пользователь уже есть, нужно просто обновить его данные.

**4. Добавление новой карточки**

Чтобы добавить на сервер новую карточку, отправьте POST-запрос:

Скопировать код

POST https://mesto.nomoreparties.co/v1/cohortId/cards

В заголовках запроса, кроме токена, необходимо отправить Content-Type, а в теле — JSON с двумя свойствами — name и link. В name должно быть название создаваемой карточки, а в link — ссылка на картинку. Если запрос прошёл успешно, сервер вернёт ответ с объектом новой карточки:

Скопировать кодJSON

{

"likes": [],

"\_id": "5d1f0611d321eb4bdcd707dd",

"name": "Байкал",

"link": "https://pictures.s3.yandex.net/frontend-developer/cards-compressed/baikal.jpg",

"owner": {

"name": "Jacques Cousteau",

"about": "Sailor, researcher",

"avatar": "https://pictures.s3.yandex.net/frontend-developer/ava.jpg",

"\_id": "ef5f7423f7f5e22bef4ad607",

"cohort": "local"

},

"createdAt": "2019-07-05T08:10:57.741Z"

},

**5. Отображение количества лайков карточки**

У каждой карточки есть свойство likes — оно содержит массив пользователей, лайкнувших карточку:

Скопировать кодJSON

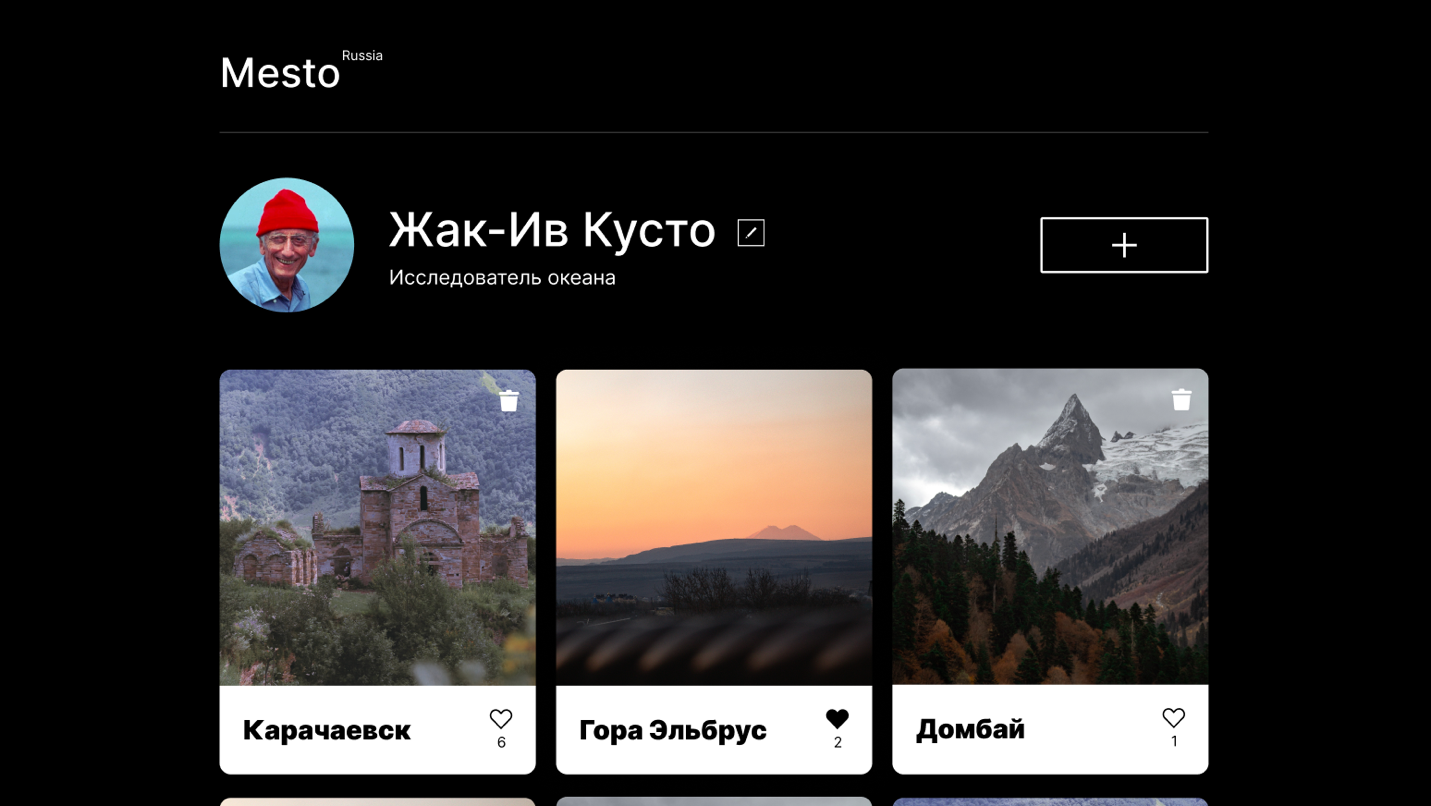
{

"likes": [],

...другие данные карточки

}

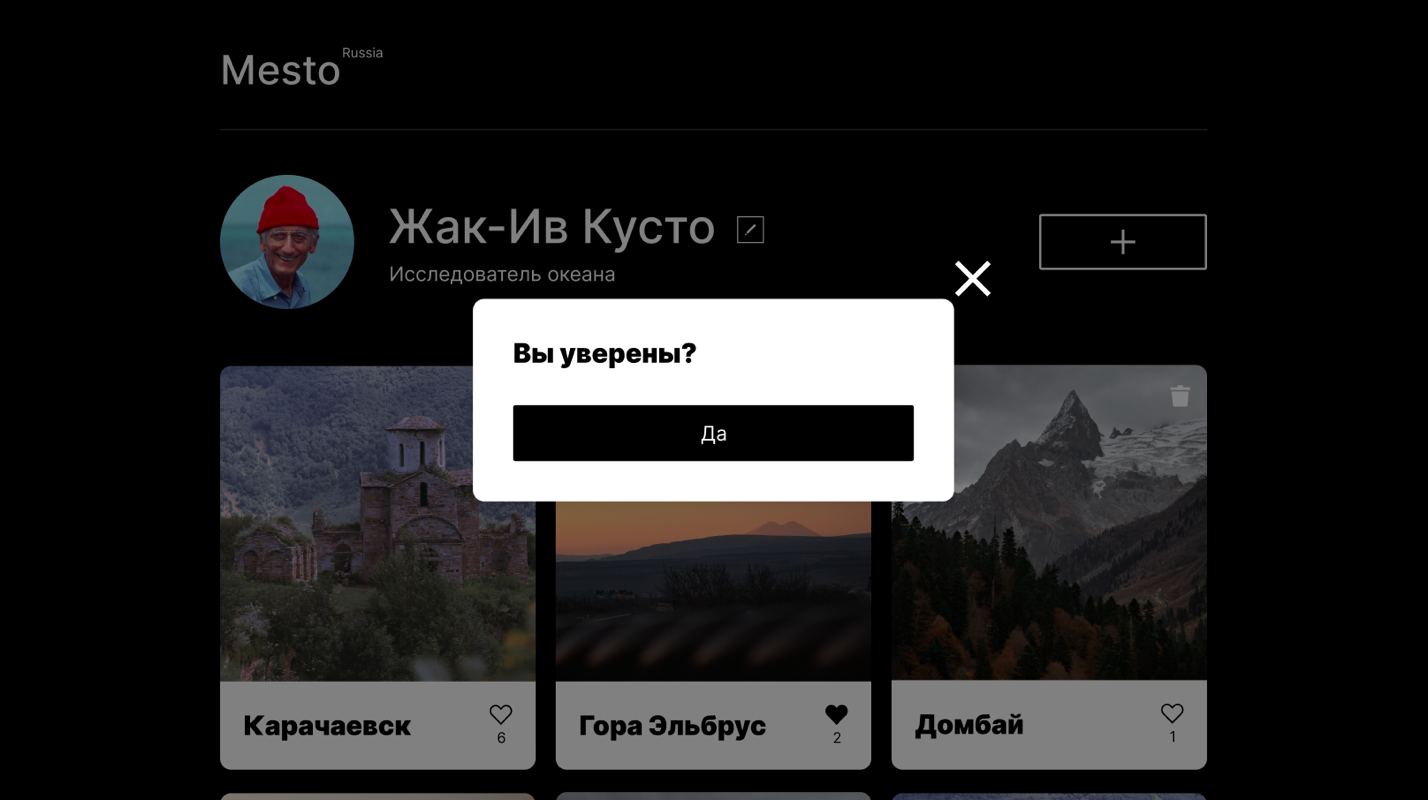
Сделайте так, чтобы на каждой карточке было написано, сколько у неё лайков:



Для этого придётся сверстать дополнительный элемент. [Его дизайн есть в Фигме](https://www.figma.com/file/PSdQFRHoxXJFs2FH8IXViF/JavaScript-9-sprint?node-id=0%3A1).

**6. Попап удаления карточки**

Удаление чего-то, как правило, безвозвратно. Поэтому перед этим действием стоит спросить пользователя, уверен ли он, что хочет удалить карточку. Для этого сделайте новый попап. Он должен открываться по клику на иконку удаления:

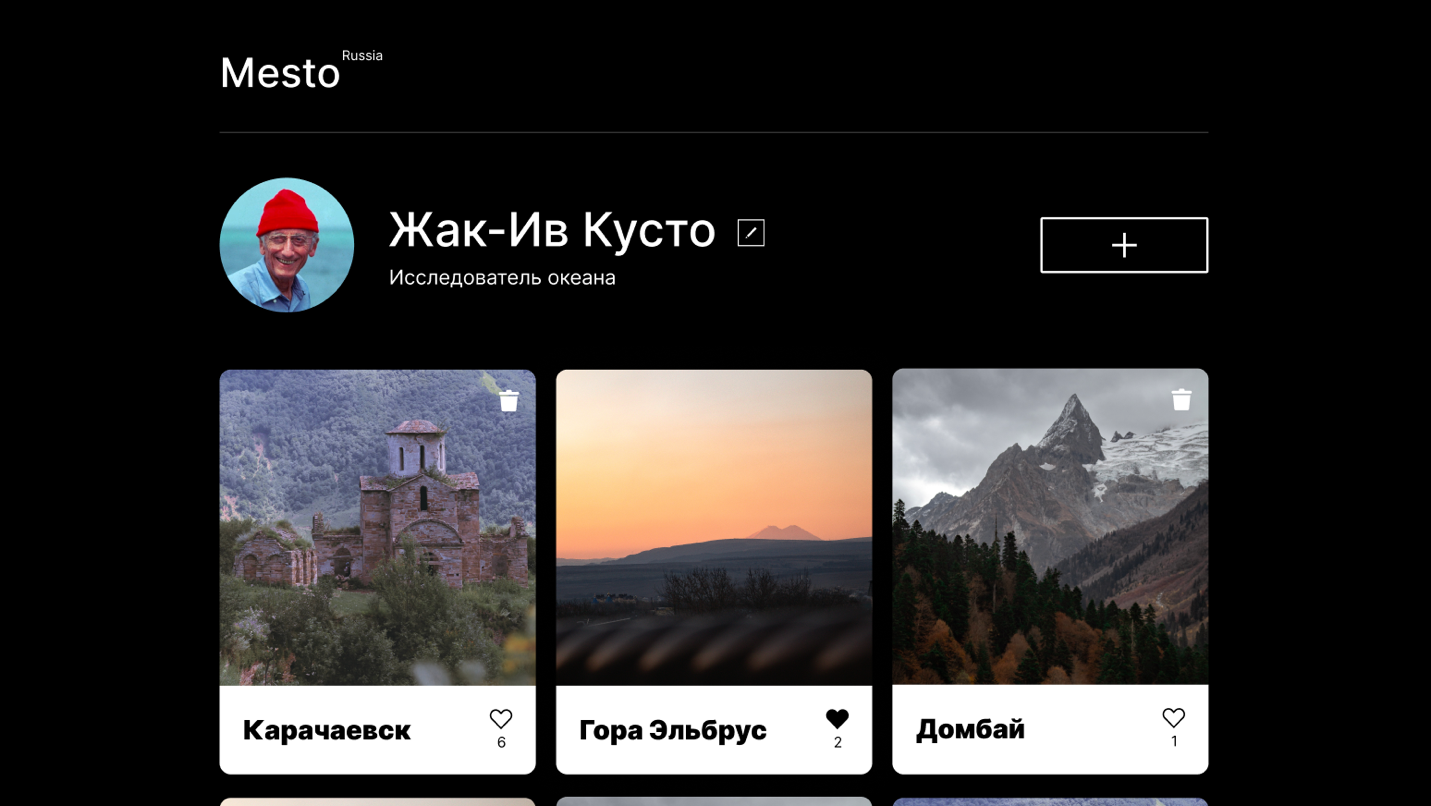


*Попап удаления карточки*

[Дизайн попапа есть в Фигме](https://www.figma.com/file/PSdQFRHoxXJFs2FH8IXViF/JavaScript-9-sprint?node-id=0%3A1).

**7. Удаление карточки**

Прежде чем браться за работу с API, исправьте элемент карточки. Сделайте так, чтобы иконка удаления была только на созданных вами карточках, так как удалять чужие карточки нельзя.



*Если карточка создана не вами, на ней нет иконки корзины*

После того, как сделаете это, реализуйте функциональность удаления карточки. Карточка должна удаляться, если в попапе удаления карточки пользователь нажал «Да».

Чтобы удалить карточку, отправьте DELETE-запрос:

Скопировать код

DELETE https://mesto.nomoreparties.co/v1/cohortId/cards/cardId

Вместо cardId в URL нужно подставить параметр \_id карточки, которую нужно удалить. \_id каждой карточки есть в её JSON:

Скопировать кодJSON

{

"likes": [],

"\_id": "5d1f0611d321eb4bdcd707dd", — вот он

...другие данные карточки

}

В итоге, запрос на удаление этой карточки должен выглядеть так:

Скопировать код

DELETE https://mesto.nomoreparties.co/v1/cohortId/cards/5d1f0611d321eb4bdcd707dd

**8. Постановка и снятие лайка**

Чтобы лайкнуть карточку, отправьте PUT-запрос:

Скопировать код

PUT https://mesto.nomoreparties.co/v1/cohortId/cards/cardId/likes

Чтобы убрать лайк, нужно отправить DELETE-запрос с тем же URL:

Скопировать код

DELETE https://mesto.nomoreparties.co/v1/cohortId/cards/cardId/likes

Вместо cardId в URL нужно подставить свойство \_id соответствующей карточки.

В ответе придёт обновлённый JSON с карточкой. Массив лайков в нём будет уже обновлён.

При постановке и снятии лайка сердечко должно менять цвет, а счётчик лайков увеличиваться или уменьшаться.

Чтобы изменить количество лайков, нужно отправить на сервер запрос с соответствующим методом. Рекомендуем брать количество лайков из ответа сервера, а не из вёрстки. Иначе могут возникнуть ошибки, когда два пользователя одновременно лайкнут одну карточку.

**9. Обновление аватара пользователя**

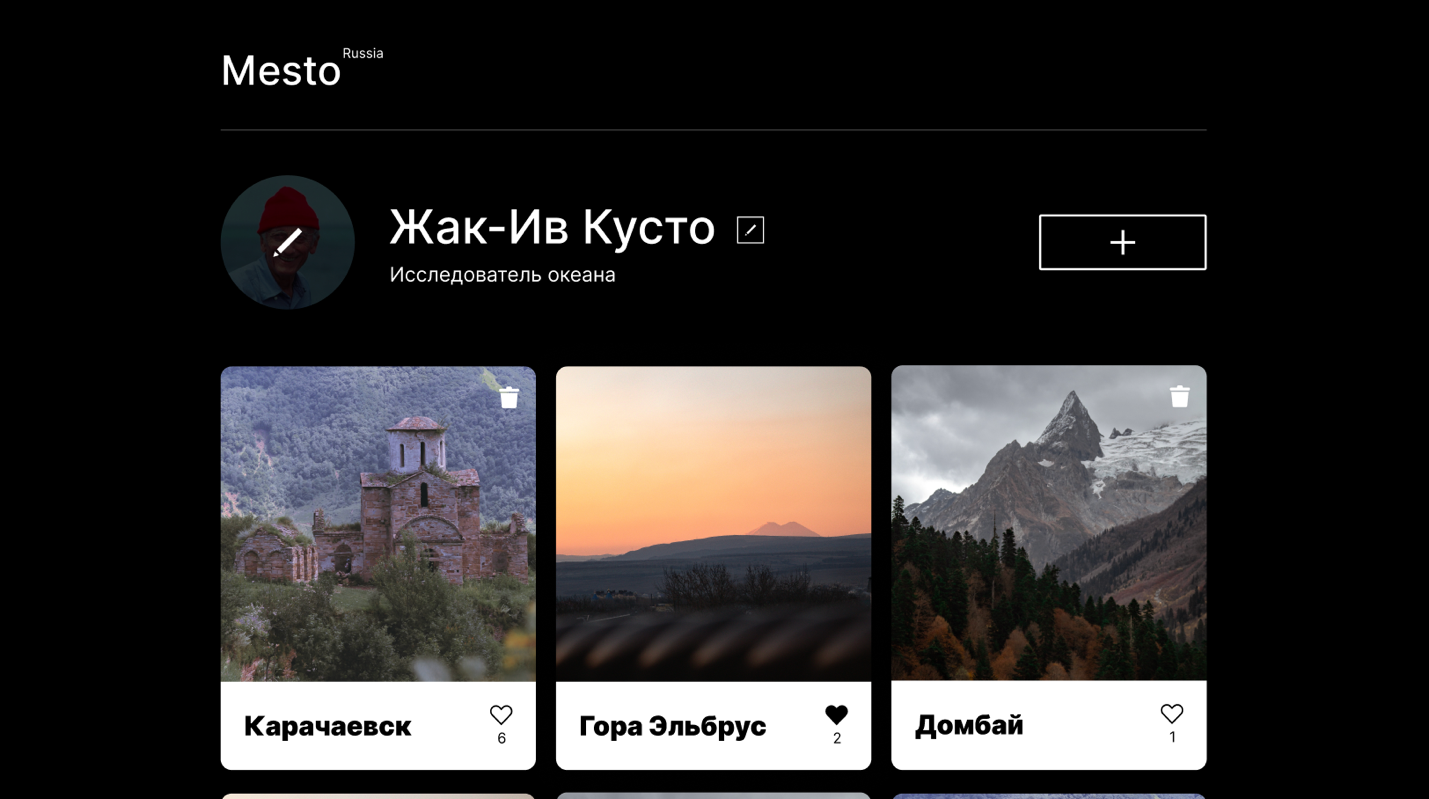
Чтобы сменить аватар, отправьте такой PATCH-запрос:

Скопировать код

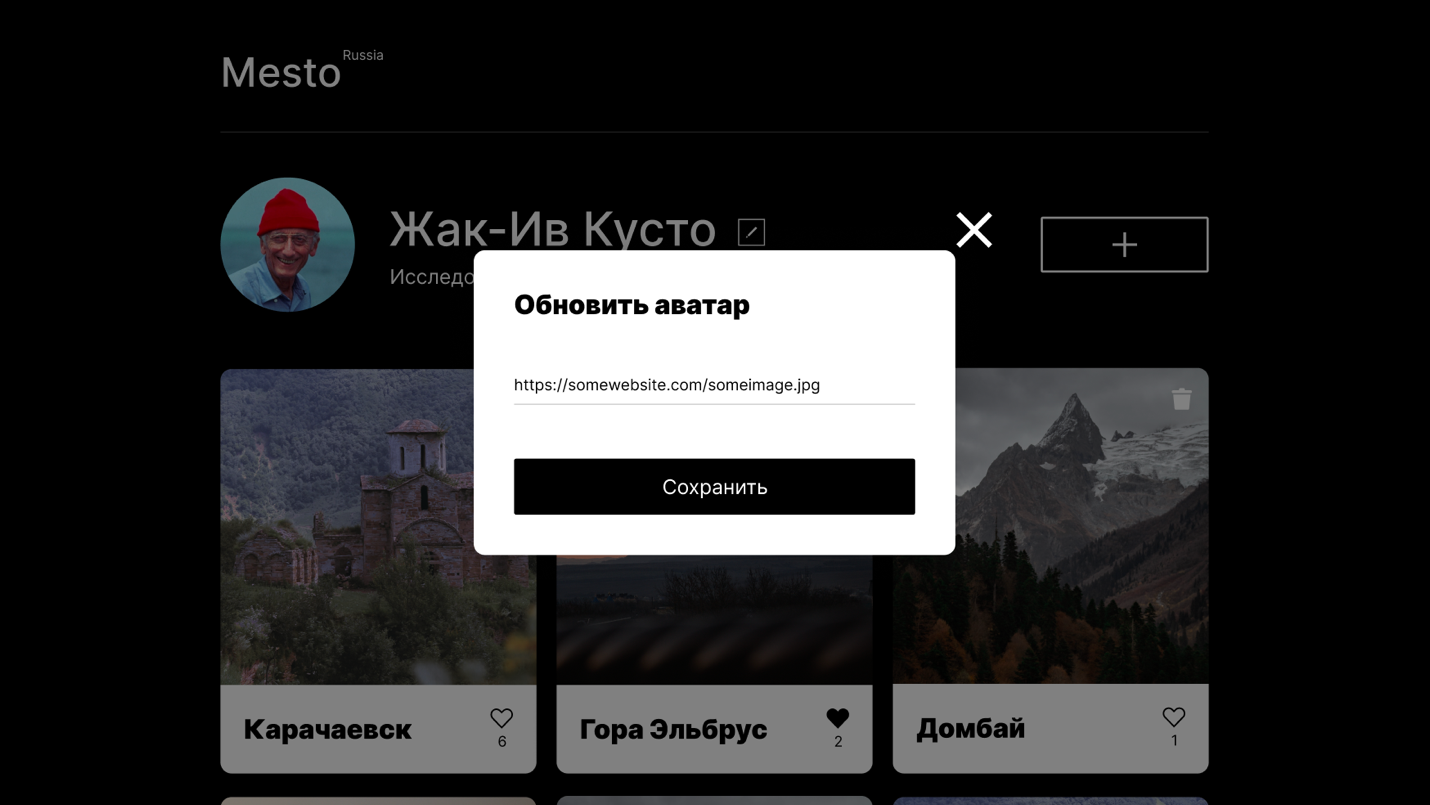
PATCH https://mesto.nomoreparties.co/v1/cohortId/users/me/avatar

В теле запроса передайте JSON с единственным свойством — avatar. Это свойство должно хранить ссылку на новый аватар. Если отправить не ссылку, сервер вернёт ошибку.

При наведении указателя мыши на аватар, на нём должна появляться иконка редактирования:



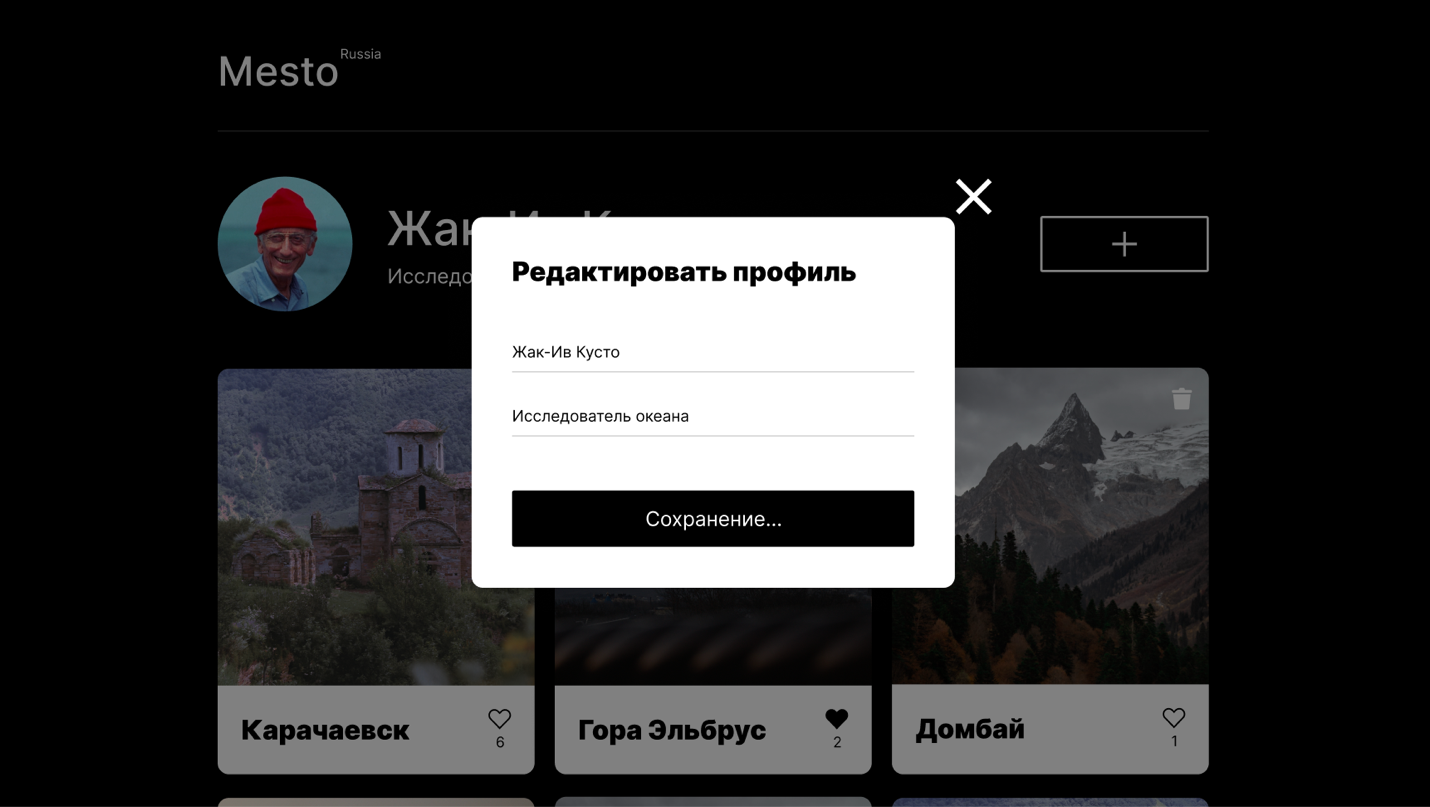
А при клике, открываться форма. Эту форму нужно сделать. В ней должно быть одно поле — ссылка на новый аватар:



Иконка редактирования аватара и форма загрузки [есть в Фигме](https://www.figma.com/file/PSdQFRHoxXJFs2FH8IXViF/JavaScript-9-sprint?node-id=0%3A1).

**10. Улучшенный UX всех форм**

Поработайте над UX. При редактировании профиля уведомите пользователя о процессе загрузки, поменяв текст кнопки на: «Сохранение...», пока данные загружаются:



Сделайте то же самое для формы добавления новой карточки и обновления аватара.

**Требования к коду**

Для работы с API создайте класс Api. Все запросы должны быть методами этого класса:

Скопировать кодJAVASCRIPT

class Api {

constructor(options) {

*// тело конструктора*

}

getInitialCards() {

*// ...*

}

*// другие методы работы с API*

}

const api = new Api({

baseUrl: 'https://mesto.nomoreparties.co/v1/cohort-42',

headers: {

authorization: 'c56e30dc-2883-4270-a59e-b2f7bae969c6',

'Content-Type': 'application/json'

}

});

**Общие комментарии**

**1. Не забывайте проверять, всё ли в порядке с ответом.** Можно использовать res.ok или res.status:

Скопировать кодJAVASCRIPT

getInitialCards() {

return fetch('https://mesto.nomoreparties.co/v1/cohort-42/cards', {

headers: {

authorization: 'c56e30dc-2883-4270-a59e-b2f7bae969c6'

}

})

.then(res => {

if (res.ok) {

return res.json();

}

});

}

**2. Учитывайте случай, когда сервер вернул ошибку.**

Скопировать кодJAVASCRIPT

getInitialCards() {

return fetch('https://mesto.nomoreparties.co/v1/cohort-42/cards', {

headers: {

authorization: 'c56e30dc-2883-4270-a59e-b2f7bae969c6'

}

})

.then(res => {

if (res.ok) {

return res.json();

}

*// если ошибка, отклоняем промис*

return Promise.reject(`Ошибка: ${res.status}`);

});

}

**3. Обрабатывайте ошибки, попадающие в catch.** Если запрос не ушёл на сервер, или тот не ответил, сработает блок catch. Обрабатывайте ошибку внутри этого блока. Если нет времени писать сложную логику, хотя бы просто выведите ошибку в консоль.

Скопировать кодJAVASCRIPT

api.getInitialCards()

.then((result) => {

*// обрабатываем результат*

})

.catch((err) => {

console.log(err); *// выведем ошибку в консоль*

});

**4. Пользуйтесь вкладкой Network для просмотра запросов.** При отправке запросов держите вкладку Network открытой. Отфильтруйте в ней XHR запросы. Это позволит оперативно следить, что приходит в ответе от сервера.