Сценарий 1. Пользователь заходит на сайт и размещает информацию о новом питомце с одновременной загрузкой фотографии

Предусловие	В системе создан пользователь с ником test_user и паролем P@ssw0rd
Шаги для повторения	1. Выполнить запрос **GET /api/key** с корректными данными пользователя. Получаем ключ для пользователя. 2. Выполнить запрос создания питомца — **POST /api/pets** с корректными данными (name, animal type, age) и фотографией (формат JPG, JPEG or PNG).
Ожидание	В ответе на запрос **GET /api/pets** с фильтром **my_pets** приходит созданный питомец с корректно загруженным файлом из шага 2 и данными питомца.

Сценарий 2. Пользователь заходит на сайт, чтобы изменить возраст питомца

Предусловие	В системе создан пользователь с ником test_user и паролем P@ssw0rd
Шаги для повторения	1. Выполнить запрос **GET /api/key** с корректными данными пользователя. Получаем ключ для пользователя 2. Выполнить запрос **GET /api/pets** с фильтром **my_pets** и выбрать первого питомца из списка, скопировать его id 3. Выполнить запрос **PUT /api/pets/{pet_id}** с параметром строки pet_id, корректным ключом авторизации, в теле запроса передать новое значение age
Ожидание	В ответе на запрос **GET /api/pets** с фильтром **my_pets** у питомца с искомым pet_id поле age изменилось на новое значение из шага 3. Поля name и animal_type остались неизменными из запроса в шаге 2.

Сценарий 3. Пользователь заходит на сайт, чтобы заменить фотографию питомца

Предусловие	В системе создан пользователь с ником test_user и паролем P@ssw0rd
Шаги для повторения	1. Выполнить запрос **GET /api/key** с корректными данными пользователя. Получаем ключ для пользователя 2. Выполнить запрос **GET /api/pets** с фильтром **my_pets** и выбрать первого питомца из списка, скопировать его id 3. Выполнить загрузку фотографии — **POST /api/pets/set_photo/{pet_id}** с корректными параметрами (параметр строки pet_id, параметр заголовка auth_key) и фотографией в формате JPG
Ожидание	В ответе на запрос **GET /api/pets** с фильтром **my_pets** у питомца с искомым pet_id изменилось поле pet_photo на загруженную фотографию из шага 3, остальные данные питомца остались неизменными из шага 2.

Сценарий 4. Пользователь пытается зайти на сайт с неправильным паролем

Предусловие	В системе создан пользователь с ником test_user и паролем P@ssw0rd
Шаги для повторения	1. Выполнить запрос **GET /api/key** с корректным email и некорректным паролем пользователя. Получаем понятное сообщение об ошибке в HTML
Ожидание	Запрос **GET /api/key** возвращает ошибку 403, тело ответа в формате HTML с описанием ошибки, в ответе отсутствует ключ авторизации (key)

Сценарий 5. Пользователь заходит на сайт для удаления питомца

Предусловие	В системе создан пользователь с ником test_user и паролем P@ssw0rd
Шаги для повторения	1. Выполнить запрос **GET /api/key** с корректными данными пользователя. Получаем ключ для пользователя 2. Выполнить запрос **GET /api/pets** с фильтром **my_pets** и выбрать первого питомца из списка, скопировать его id 3. Выполнить запрос **DELETE api/pets/{pet_id}** с корректным ключом авторизации и параметром строки pet_id
Ожидание	В ответе на запрос **GET /api/pets** с фильтром **my_pets** отсутствует питомец с удаленным pet_id

Тест кейсы для API Дом питомца

GET /api/key

Вызов API	Результат получения запроса

Базовый позитивный сценарий
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: корректный email
password: корректный пароль
	• Код ответа 200 (OK)
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin : *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: application/json; server: gunicorn/20.0.4; via: 1.1 vegur
	• В теле приходит json с корректной структурой
	• В теле приходит следующее поле:
	  ○ key: <auth_key>
	• Время выполнения: не дольше 1с

Негативный тест: корректный email, некорректный пароль
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: корректный email
password: некорректный пароль
	• Код ответа 403 (Forbidden)
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin: *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: text/html; charset=utf-8; server: gunicorn/20.0.4; via: 1.1 vegur
	• В теле приходит html со следующим содержимым:
	
	*<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
	<title>403 Forbidden</title>
	 Forbidden
	 This user wasn't found in database *
	
	• Время выполнения: не дольше 1с

Негативный тест: некорректный email, корректный пароль
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: корректный email
password: некорректный пароль
	• Код ответа 403 (Forbidden)
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin: *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: text/html; charset=utf-8; server: gunicorn/20.0.4; via: 1.1 vegur
	• В теле приходит html со следующим содержимым:
	
	*<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
	<title>403 Forbidden</title>
	 Forbidden
	 This user wasn't found in database *
	
	Время выполнения: не дольше 1с

Негативный тест: корректный email, пустое поле ввода пароля
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: корректный email
password: <blank>
	• Код ответа 403 (Forbidden)
	
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin: *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: text/html; charset=utf-8; server: gunicorn/20.0.4; via: 1.1 vegur
	
	• В теле приходит html со следующим содержимым:
	
	*<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
	<title>403 Forbidden</title>
	 Forbidden
	 This user wasn't found in database *
	
	Время выполнения: не дольше 1с

Негативный тест: корректный email, пароль введен ALL CAPS
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: корректный email
password: корректный password написан ALL CAPS
	• Код ответа 403 (Forbidden)
	
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin: *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: text/html; charset=utf-8; server: gunicorn/20.0.4; via: 1.1 vegur
	
	• В теле приходит html со следующим содержимым:
	
	*<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
	<title>403 Forbidden</title>
	 Forbidden
	 This user wasn't found in database *
	
	Время выполнения: не дольше 1с

Позитивный тест: корректный email введен ALL CAPS, корректный пароль
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: корректный email введен ALL CAPS
password: корректный password
	• Код ответа 200 (OK)
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin: *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: application/json; server: gunicorn/20.0.4; via: 1.1 vegur
	• В теле приходит json с корректной структурой
	• В теле приходит следующее поле:
	  ○ key: <auth_key>
	Время выполнения: не дольше 1с

Негативный тест: некорректное тип данных в значении email, корректный пароль
 	
GET
https://petfriends1.herokuapp.com/api/key
 	
Headers:
email: 12345678
password: некорректный пароль
	• Код ответа 403 (Forbidden)
	• Заголовки ответа содержат:
	  ○ Обязательно: access-control-allow-origin: *; date: <date>
	  ○ Опционально: connection: keep-alive; content-type: text/html; charset=utf-8; server: gunicorn/20.0.4; via: 1.1 vegur
	• В теле приходит html со следующим содержимым:
	
	*<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
	<title>403 Forbidden</title>
	 Forbidden
	 This user wasn't found in database *
	
	• Время выполнения: не дольше 1с

POST /api/pets/

Вызов API	Результат получения запроса

Базовый позитивный сценарий
 	
POST
https://petfriends1.herokuapp.com/api/pets/
 	
Headers:
 	
auth_key: Корректный ключ API
 	
Body:
name: имя питомца, string
animal_type: тип животного, string
age: возраст, number
pet_photo: корректный файл JPG, JPEG or PNG
	• Код ответа 200 (Created)
	• В теле приходит json с корректной структурой*
	• В теле приходят следующие поля в объектах pets:
	  ○ age: целое число,
	  ○ animal_type: строка,
	  ○ created_at: строка со снимком времени (timestamp)
	  ○ id: uuid
	  ○ name: строка
	  ○ pet_photo: base64 image
	  ○ user_id: guid
	• Заголовки ответа:
	
	Обязательно:
	  ○ content-type: content-type: text/html; charset=utf-8
	  ○ date: <дата>
	
	Опционально:
	  ○ connection: keep-alive
	  ○ server: gunicorn/20.0.4
	  ○ via: 1.1 vegur
	
	Время выполнения: не дольше 1с
	
	**Примечание: по факту выполнения запроса ответ пришел в формате html, а не json (как в документации), хотя запрос прошел со статусом 200 (OK) и питомец был успешно добавлен*

Негативный тест: некорректный ключ авторизации (auth_key) **POST** https://petfriends1.herokuapp.com/api/pets/ **Headers:** auth_key: 1234ab **Body:** name: имя питомца, string animal_type: тип животного, string age: возраст, number pet_photo: корректный файл JPG, JPEG or PNG	• Код ответа 403 (auth_key is incorrect)* • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с **Примечание: по факту выполнения даже с некорректным auth_key происходит добавление питомца и запрос возвращает код 200 (OK). Это баг уязвимости системы – используя стороннее ПО любой человек может добавить питомца без авторизации*
Негативный тест: некорректный файл вместо фото **POST** https://petfriends1.herokuapp.com/api/pets/ **Headers:** auth_key: Корректный ключ API **Body:** name: имя питомца, string animal_type: тип животного, string age: возраст, number pet_photo: текстовый файл txt	• Код ответа 400 (Bad Request)* • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с **Примечание: согласно документации мы должны получить ошибку 400, по факту – получаем ошибку 500 Error: INTERNAL SERVER ERROR*
Негативный тест: текстовое значение в поле возраст **POST** https://petfriends1.herokuapp.com/api/pets/ **Headers:** auth_key: Корректный ключ API **Body:** name: имя питомца, string animal_type: тип животного, string age: txt pet_photo: корректный файл JPG, JPEG or PNG	• Код ответа 400 (Bad Request)* • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с

Негативный тест: отсутствующие обязательные параметры **POST** https://petfriends1.herokuapp.com/api/pets/ **Headers:** auth_key: Корректный ключ API **Body:** name: <blank> animal_type: <blank> age: <blank> pet_photo: корректный файл JPG, JPEG or PNG	• Код ответа 400 (Bad Request)* • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с **Примечание: по факту выполнения запрос проходит (статус 200) с использованием Python, и позволяет добавить питомца с пустыми параметрами. Это баг.*
Негативный тест: слишком большое значение в поле age **POST** https://petfriends1.herokuapp.com/api/pets/ **Headers:** auth_key: Корректный ключ API **Body:** name: имя питомца, string animal_type: тип животного, string age: 12678000001 pet_photo: корректный файл JPG, JPEG or PNG	• Код ответа 400 (Bad Request)* • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с **Примечание: по факту выполнения запрос проходит (статус 200), хотя значение возраста 12678000001 не имеет смысла с точки зрения бизнес-логики*

PUT /api/pets/{pet_id}

Вызов API	Результат получения запроса

Базовый позитивный сценарий
 	
PUT
https://petfriends1.herokuapp.com/api/pets/{pet_id}
 	
Headers:
 	
auth_key: Корректный ключ API
 	
Path_params:
pet_id=корректный id питомца
 	
Body:
name: новое имя питомца, string
animal_type: новый тип животного, string
age: новый возраст, number
	• Код ответа 200 (Updated)
	• В теле приходит json с корректной структурой*
	• В теле приходят следующие поля:
	  ○ age: целое число,
	  ○ animal_type: строка,
	  ○ created_at: строка со снимком времени (timestamp)
	  ○ id: uuid (идентично отправляемому pet_id в запросе)
	  ○ name: строка
	  ○ pet_photo: base64 image
	  ○ user_id: guid
	
	• Заголовки ответа:
	
	Обязательно:
	○ content-type: application/json
	○ date: <дата>
	
	Опционально:
	○ connection: keep-alive
	○ server: gunicorn/20.0.4
	○ via: 1.1 vegur
	
	Время выполнения: не дольше 1с

Позитивный сценарий: из трех опциональных параметров запроса задан только параметр name
 	
PUT
https://petfriends1.herokuapp.com/api/pets/{pet_id}
 	
Headers:
 	
auth_key: Корректный ключ API
 	
Path_params:
pet_id=корректный id питомца
 	
Body:
name: новое имя питомца, string
animal_type: <blank>
age: <blank>
	• Код ответа 200 (Updated)
	• В теле приходит json с корректной структурой*
	• В теле приходят следующие поля:
	  ○ age: целое число,
	  ○ animal_type: строка,
	  ○ created_at: строка со снимком времени (timestamp)
	  ○ id: uuid (идентично отправляемому pet_id в запросе)
	  ○ name: строка
	  ○ pet_photo: base64 image
	  ○ user_id: guid
	
	• Заголовки ответа:
	
	Обязательно:
	○ content-type: application/json
	○ date: <дата>
	
	Опционально:
	○ connection: keep-alive
	○ server: gunicorn/20.0.4
	○ via: 1.1 vegur
	
	Время выполнения: не дольше 1с
	
	**Примечание: ожидается, что при пустых опциональных параметрах в запросе (animal_type, age) останутся предыдущие значения. Вместо этого они меняются на значение “None”. Это баг.*

Позитивный сценарий: из трех опциональных параметров запроса задан только параметр animal_type
 	
PUT
https://petfriends1.herokuapp.com/api/pets/{pet_id}
 	
Headers:
 	
auth_key: Корректный ключ API
 	
Path_params:
pet_id=корректный id питомца
 	
Body:
name: <blank>
animal_type: тип питомца, string
age: <blank>
	• Код ответа 200 (Updated)
	• В теле приходит json с корректной структурой*
	• В теле приходят следующие поля:
	  ○ age: целое число,
	  ○ animal_type: строка,
	  ○ created_at: строка со снимком времени (timestamp)
	  ○ id: uuid (идентично отправляемому pet_id в запросе)
	  ○ name: строка
	  ○ pet_photo: base64 image
	  ○ user_id: guid
	
	• Заголовки ответа:
	
	Обязательно:
	○ content-type: application/json
	○ date: <дата>
	
	Опционально:
	○ connection: keep-alive
	○ server: gunicorn/20.0.4
	○ via: 1.1 vegur
	
	Время выполнения: не дольше 1с
	
	**Примечание: ожидается, что при пустых опциональных параметрах в запросе (name, age) останутся предыдущие значения. Вместо этого они меняются на значение “None”. Это баг.*

Позитивный сценарий: из трех опциональных параметров запроса задан только параметр age
 	
PUT
https://petfriends1.herokuapp.com/api/pets/{pet_id}
 	
Headers:
 	
auth_key: Корректный ключ API
 	
Path_params:
pet_id=корректный id питомца
 	
Body:
name: <blank>
animal_type: <blank>
age: возраст, number
	• Код ответа 200 (Updated)
	• В теле приходит json с корректной структурой*
	• В теле приходят следующие поля:
	  ○ age: целое число,
	  ○ animal_type: строка,
	  ○ created_at: строка со снимком времени (timestamp)
	  ○ id: uuid (идентично отправляемому pet_id в запросе)
	  ○ name: строка
	  ○ pet_photo: base64 image
	  ○ user_id: guid
	
	• Заголовки ответа:
	
	Обязательно:
	○ content-type: application/json
	○ date: <дата>
	
	Опционально:
	○ connection: keep-alive
	○ server: gunicorn/20.0.4
	○ via: 1.1 vegur
	
	Время выполнения: не дольше 1с

	**Примечание: ожидается, что при пустых опциональных параметрах в запросе (name, animal_type) останутся предыдущие значения. Вместо этого они меняются на значение "None". Это баг.*
Негативный сценарий: некорректный pet_id **PUT** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: Корректный ключ API **Path_params:** pet_id=некорректный id питомца **Body:** name: новое имя питомца, string animal_type: новый тип животного, string age: новый возраст, number	• Код ответа 400 (bad request) • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с
Негативный сценарий: некорректный auth_key **PUT** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: некорректный ключ API (123ab) **Path_params:** pet_id=корректный id питомца **Body:** name: новое имя питомца, string animal_type: новый тип животного, string age: новый возраст, number	• Код ответа 403 (auth_key is incorrect) • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur **Примечание: согласно документации, ожидаем код 403, однако по факту получаем код 200 (OK) с изменением информации о питомце*
Негативный сценарий: спецсимволы и кириллические символы ввода **PUT** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: корректный ключ API **Path_params:** pet_id=корректный id питомца	• Код ответа 400 (incorrect data) • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с

Body: name: *";№!!"%*№"+"!))_,ъясй* animal_type: новый тип животного, string age: новый возраст, number	**Примечание: ожидаем код 400, однако по факту получаем код 200 (OK) с изменением информации о питомце = несоответствие бизнес-логике*
Негативный сценарий: строка вместо числа в поле возраст **PUT** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: корректный ключ API **Path_params:** pet_id=корректный id питомца **Body:** name: новое имя питомца, string animal_type: новый тип животного, string age: qwertyйцукен	• Код ответа 400 (bad request) • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с
Негативный тест: слишком длинное поле ввода (больше 256 символов) в поле «Имя» **PUT** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: корректный ключ API **Path_params:** pet_id=корректный id питомца **Body:** name: новое имя питомца, string > 256 символов animal_type: новый тип животного, string age: qwertyйцукен	• Код ответа 400 (bad request) • В теле ответа HTML код с сообщением о некорректном запросе • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с **Примечание: ожидаем код 400, однако по факту получаем код 200 (OK) с изменением информации о питомце – несоответствие бизнес-логике*
Негативный сценарий: изменение питомца, не соответствующему авторизированному пользователю **PUT** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: корректный ключ API **Path_params:**	• В документации данный случай не предусмотрен, должна быть предусмотрена ошибка авторизации, так как пользователь должен иметь возможность изменять только своих питомцев, однако возвращается статус 200 (OK) и данные питомца изменяются • В результате имеется уязвимость системы, когда любой пользователь с корректным кодом авторизации может изменять всех животных на сайте

pet_id=корректный id питомца **Body:** name: новое имя питомца, string animal_type: новый тип животного, string age: новый возраст, number	

DELETE /api/pets/{pet_id}

Вызов API	Результат получения запроса
Базовый позитивный сценарий **DELETE** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: Корректный ключ API **Path_params:** pet_id=корректный id питомца не входящий в список my_pets	• Код ответа 200 (OK) • В теле ответа HTML с обновленным списком питомцев, в котором отсутствует удаленный pet_id • Заголовки ответа Обязательно: ○ content-type:text/html charset=utf-8 ○ date: <дата> Опционально: ○ connection: keep-alive ○ server: gunicorn/20.0.4 ○ via: 1.1 vegur • Время выполнения: не дольше 1с
Негативный сценарий: повторное удаление питомца **DELETE** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: Корректный ключ API **Path_params:** pet_id = id недавно удаленного питомца	• В документации данный случай не предусмотрен, должна быть предусмотрена ошибка 400 (bad request), однако возвращается статус 200 (OK). В результате отсутствует возможность проверки того, правильно ли был задан pet_id в запросе на удаление
Негативный сценарий: некорректный pet_id **DELETE** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: Корректный ключ API **Path_params:** pet_id = 123abc	• В документации данный случай не предусмотрен, должна быть предусмотрена ошибка 400 (bad request), однако возвращается статус 200 (OK). В результате отсутствует возможность проверки того, правильно ли был задан pet_id в запросе на удаление

Негативный сценарий: некорректный auth_key
 	
DELETE
https://petfriends1.herokuapp.com/api/pets/{pet_id}
 	
Headers:
 	
auth_key: некорректный ключ API (123abc)
 	
Path_params:
pet_id=корректный id питомца
	• Код ответа 403 (auth_key is incorrect)*
	• В теле ответа HTML с обновленным списком питомцев, в котором отсутствует удаленный pet_id
	• Заголовки ответа
	Обязательно:
	  ○ content-type:text/html charset=utf-8
	  ○ date: <дата>
	Опционально:
	  ○ connection: keep-alive
	  ○ server: gunicorn/20.0.4
	  ○ via: 1.1 vegur
	• Время выполнения: не дольше 1с
	
	**Примечание: по факту выполнения запроса статус 200 (OK) = уязвимость системы, так как возможно удаление питомца при неправильном ключе авторизации*

Негативный сценарий: удаление питомца, не соответствующему авторизированному пользователю **DELETE** https://petfriends1.herokuapp.com/api/pets/{pet_id} **Headers:** auth_key: Корректный ключ API **Path_params:** pet_id=корректный id питомца не входящий в список my_pets	• В документации данный случай не предусмотрен, должна быть предусмотрена ошибка авторизации, так как пользователь должен иметь возможность удалять/изменять только своих животных однако возвращается статус 200 (OK) и животное удаляется • В результате имеется уязвимость системы, когда любой пользователь с корректным кодом авторизации может удалять всех животных на сайте