Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Отчет по учебному курсу «Методы тестирования программного обеспечения»

Тестирование мобильного сервиса «PetsSearching»

Выполнила:
студентка 4 курса группы 22407
Е. С. Жданович

Петрозаводск — 2020

Содержание

1 Объект тестирования

В качестве объекта тестирования выступает мобильный сервис PetsSearching. Он позволяет публиковать и просматривать объявления о пропаже домашних животных, а также связываться с хозяином пропавшего животного в чате.

Приложение разработано на платформе Android. Использовался язык разметки xml и язык программирования Java. Серверное приложение разработано на языке NodeJS с использованием Express.js и базы данных MongoDB.

Возможности серверной части:

1. получать запросы от клиентов и интерпретировать их назначение;
2. выполнять обработку операций согласно логике запроса;
3. взаимодействовать с базами данных для хранения информации и обеспечения доступа к ней;
4. отправлять ответ клиенту на его запрос с указанием статуса и результата запрос.

Возможности клиентской части:

1. предоставлять интерфейс для взаимодействия пользователя и информации;
2. принимать информацию от пользователя;
3. формировать заголовок и тело запроса, и отправлять данные на сервер;
4. принимать ответы от сервера и обрабатывать полученную информацию.

Основные функции приложения:

1. Вход через google-аккаунт с использованием firebase authentication.
2. Добавление пропавшего животного (текстовая информация –> основная фотография и дополнительные фотографии –> отметка на карте).
3. Просмотр списка пропавших животных.
4. Просмотр карты с координатами всех пропавших животных.
5. Просмотр страницы одного пропавшего животного.
6. Сохранение животных в отдельный список «избранные животные», просмотр этого списка.

7. Чат с владельцем пропавшего животного.

8. Просмотр списка животных, добавленных пользователем.

9. Просмотр архива объявлений пользователя (животные уже были найдены).

10. Удаление объявления.

2 Стратегия тестирования

2.1 Архитектура приложения

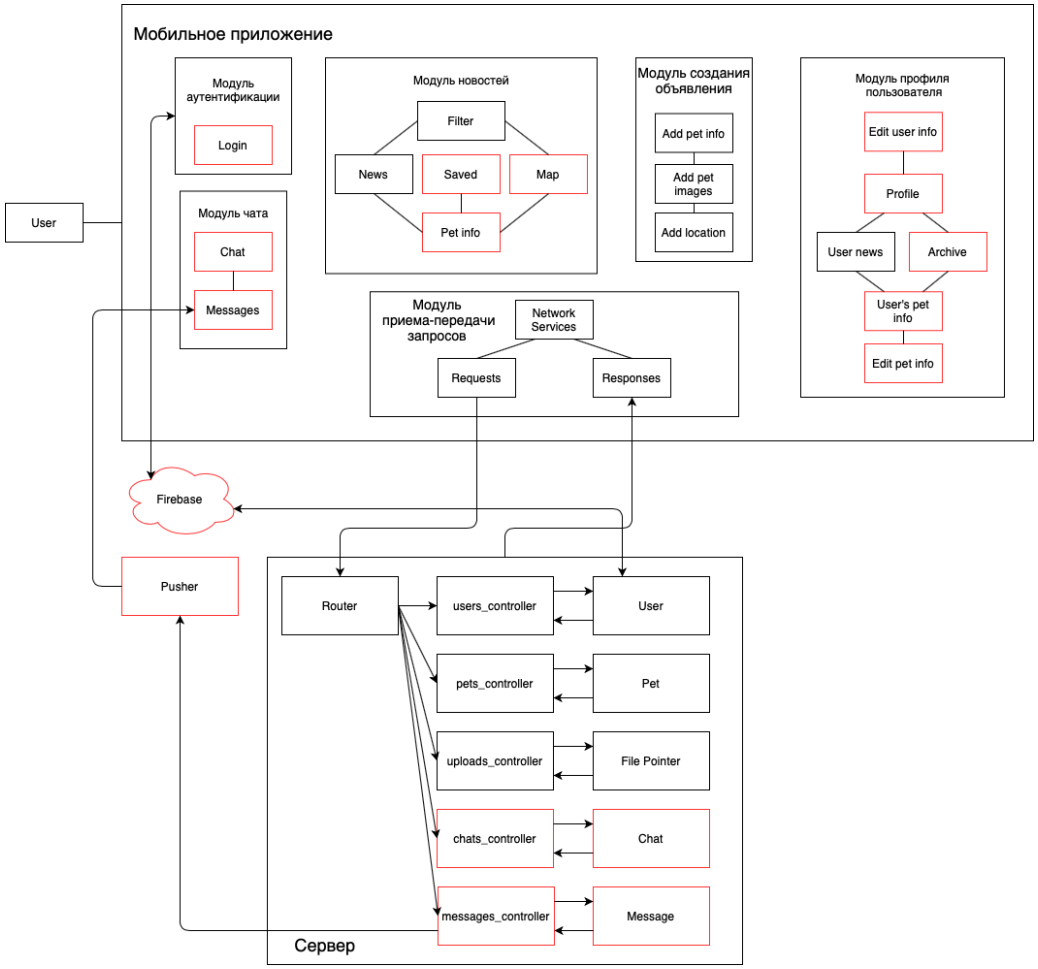

Рис. 1: Архитектра приложения

Архитектура frontend-приложения состоит из нескольких модулей:

1. модуль аутентификации отвечает за вход и регистрацию пользователя в системе. Аутентификация осуществляется через Google-аккаунт с помощью Firebase. При регистрации или входе для пользователя создается уникальный токен доступа, который идентифицирует его при отправке запросов на сервер;

2. модуль новостей отвечает за отображение объявлений пропавших животных как в виде списка, так и на карте. Имеется возможность фильтровать новости по следующим параметрам: порода, цвет, возраст, пол. Модуль также предоставляет возможность просмотра конкретного объявления и просмотра закладок пользователя;

3. модуль создания объявления о пропаже животного;

4. модуль чата отвечает за возможность коммуникации между пользователями в режиме реального времени. Для отслеживания приема-отправки сообщений используется технология Pusher — пользователь подписывается на получение сообщений от приватного канала, создаваемого для каждого чата;

5. модуль профиля пользователя включает в себя следующие возможности: просмотр и редактирование информации профиля, просмотр активных (не найденных) объявлений и их редактирование, просмотр архива объявлений;

6. модуль клиент-серверного взаимодействия. Для его реализации была использована библиотека Ion. В целях структурирования кода, для каждой функции, осуществляющей прием и передачу данных, созданы классы запроса и ответа. Эти функции объединены в сервисы по тому к какому контроллеру на сервере они обращаются. Всего в рамках проекта реализованы следующие сервисы: UsersService, PetsService, ChatsService, MessagesService.

Архитектура backend-приложения построена в рамках архитектуры MVC:

1. router — механизм распределения маршрута, определяет какой контроллер должен обрабатывать запрос;

2. controller — принимает запрос и формирует ответ для отправки клиенту. В серверном приложении реализованы следующие контроллеры: users_controller, pets_controller, chats_controller, messages_controller, uploads_controller:

   (a) users_controller обрабатывает следующие запросы: аутентифкация пользователя, обновление fcm-токена пользователя для отправки на его устройство push-уведомлений, выход из системы, получение информации о пользователе, обновление информации пользователя;

   (b) pets_controller обрабатывает следующие запросы: получение новостей в виде списка по фильтру, получение новостей для карты согласно текущему местоположению пользователя(показываются ближайшие, при уменьшении масштаба карты добавляются остальные), создание объявления о пропавшем животном, получение объявлений, закладок и архива пользователя, редактирование пропавшего животного, смена статуса животного(найден/потерян), удаление объявления;

(c) chats_controller обрабатывает следующие запросы: получение чатов пользователя, создание нового чата, проверка существоания чата между пользователями;

(d) messages_controller обрабатывает следующие запросы: получение сообщений конкретного чата, подписка на приватный канал для обновления сообщений чата, создание нового сообщения и обновление сообщений участников приватного канала, обновление статуса сообщений с "не прочитано"на "прочитано";

(e) uploads_controller обрабатывает следующие запросы: загрузка одного файла, загрузка множества файлов.

3. model — предоставляют функции и методы для работы с данными. В серверном приложении реализованы следующие модели: user, pet, chat, message, filePointer:

   (a) user содержит следующие функции и методы:

      i. метод refreshToken() — обновляет токен доступа при истичении срока его достоверности;

      ii. метод refreshFcmToken(fcmToken) — добавляет fcmToken устройства пользователя (fcmToken ипользуется для отправки push-уведомлений на устройство);

      iii. метод updateObject(data = []) — обновляет поля объекта пользователя согласно информации переданной в data. В data могут содердаться поля name, gender,age, image_id;

      iv. метод exportObject() — формирует и возвращает объект, содержащий поля пользователя, необходиые для frontend-приложения: id, name, age, image_url, gender;

      v. функция createObject(data = []) — создает объект пользователя в базе данных исходи из информации переданой в data. В data содержатся firebase_user_id, name, age, gender.

   (b) pet содержит следующие функции и методы:

      i. метод exportObject() — формирует и возвращает объект, содержащий поля животного, необходимые для frontend-приложения: id, name, breed, color, age, description, longitude, latitude, image_url, secondary_images, gender, status, user, created_at;

ii. функция exportObjects(objects = []) — применяет функцию exportObject() для каждого животного в массиве objects и возвращает массив экспортированных животных;

iii. функция createObject(data = []) — создает объект животного исходи из информации переданной в data. В data должны бать следующие поля для успешного создания животного: name, breed_id, color, age, gender, description, longitude, latitude, image_id;

iv. функция findObject(data = []) — осуществляет поиск по животных по параметрам заданным в data. В data могут содержаться следующие поля: name, breed_id, color, age, gender, status. Если data пуст, то возвращаются все животные;

v. функция findMapObject(data = []) — осуществляет поиск по животных по параметрам заданным в data. В data содержаться следующие поля, определяющие турриторию для поиска животных в ней: swLat (широта юго-западной точки), neLat (широта северо-восточной точки), neLng(долгота северо-восточной точки) , swLng (долгота юго-западной точки);

vi. функция getArchived(user_id) — возвращает список животных в архиве пользователя;

vii. функция exportBookmarks(user_id) —возвращает список объектов животных, которые находятся в закладках пользователя;

viii. метод changeStatus() — изменяет статус животного на противоположный (lost/found);

ix. метод updateObject(data = []) — редактирует поля объекта животного согласно информации, содержащейся в data. В data могут содержаться следующие поля: name, breed_id, color, age, gender, description, latitude, longitude, image_url;

x. метод updateImages(imagesId) — добавляет в массив secondary_images объекта животного id изображений;

xi. метод deleteImage(imageId) — удаляет из массива secondary_images объекта животного id фотографии;

xii. метод deleteObject() — удаляет объект живтного.

(c) chat содержит следующие функции и методы:

i. метод exportObject() — формирует и возвращает объект, содержащий поля чата, необходимые для frontend-приложения: id, recipient, unread_messages, last_message, created_at;

ii. метод exportObjects(objects = []) — применяет функцию exportObject() для каждого чата в массиве objects и возвращает массив экспортированных чатов;

iii. функция createObject(users_ids = []) — создает новый объект чат с полями id пользователей этого чата.

(d) message содержит следующие функции и методы:

i. метод exportObject() — формирует и возвращает объект, содержащий поля сообщения, необходимые для frontend-приложения: id, text, chat_id, is_red, sender, created_at;

ii. метод exportObjects(objects = []) — применяет функцию exportObject() для каждого сообщения в массиве objects и возвращает массив экспортированных сообщений;

iii. функция createObject(text, user, chatId) — создает новый объект сообщение;

iv. функция getMessages(chat_id) — возвращает список сообщений чата;

v. функция setRed(chat) — устанавливает статус сообщений id_red с false на true.

(e) filePointer содержит следующие функции и методы:

i. метод exportObject() — формирует и возвращает объект, содержащий поля, необходимые для frontend-приложения: id, url;

ii. метод getUrl() — возвращает url по которому доступен файл;

iii. функция createObject(path) — создает объект в котором хранится информация о пути до файла.

4. Также написана библиотека, которая содержит дополнительные функции:

(a) authentication(req, res) — проверяет наличие пользователя, отправившего запрос, в базе данных и в случае его существования возвращает объект user, иначе ошибка и статус код 401 Unauthorized. Из заголовка запроса извлекает токен доступа access_token идентифицирующий пользователя и осуществляется поиск по нему в бд;

(b) sendPush(fcmTokens, title, body) — отправляет push-уведомление на устройство пользователя;

(c) fetchUser(req, res) — осуществляет поиск и возвращает объект пользователя по user_id, передавемому в строке url;

(d) fetchPet(req, res) — осуществляет поиск и возвращает объект животного по pet_id, передавемому в строке url;

(e) fetchChat(req, res) — осуществляет поиск и возвращает объект чата по chat_id, передавемому в строке url.

В рамках работы будут протестированы функции добавления пропавшего животного, просмотра списка пропавших животных и удаления объявления.

1. Добавление животного происходит в 3 этапа – добавление текстовой информации, добавление фотографий, добавление геопозиции. Информация передается на сервер, проверяется ее корректность. В итоге информация добавляется в таблицу Pet в базе данных, а ссылки на фотографии – в таблицу FilePointer. Используемые компоненты: pets_controller, модель pet, модуль создания объявления о пропаже животного;

2. Просмотр списка новостей. Пользователь переходит на страницу новостей. Передается запрос на сервер на получение всех новостей. В качестве ответа приходит массив новостей, которые отображаются на странице новостей. Используемые компоненты: pets_controller и модель pet;

3. Удаление новости. Пользователь переходит на страницу животного, добавленного самим пользователем и нажимает на кнопку «Удалить новость». Id собаки передается на сервер в строке url. На сервере находится объект собаки, и уничтожается. Используемые компоненты: pets_controller и модель pet;

2.2 Стратегия модульного тестирования

Для подготовки к модульному тестированию была создана отдельная база данных, идентичная той, которая используется в приложении. В новой базе данных, в таблице пользователей, было создано 2 пользователя – user1 и user2. В таблице пород была добавлена одна порода. И в таблице животных - одно животное.

Модульное тестирование позволяет проверить на корректность отдельные модули исходного кода программы. В моем случае тестируются функции API с помощью библиотеки

chai. Тесты запускаются через терминал и туда же выводится результат.

Тестироваться будут функции:

- authentication(req, res) – проверка, аутентифицирован ли пользователь, который посылает запрос.
- createObject(data = [], user) – создает пропавшее животное.
- findObjects(data = []) – возвращает список животных по заданным параметрам.
- deleteObject(petId) – удаляет животное.

Функции, которые не будут тестироваться:

- exportObjects(objects = []) — принимает массив объектов класса Pet и возвращает их со значениями, необхдимыми для клиентского приложения;
- exportObject() — экспортирует один объект класса Pet.

Для каждой из функций приложения (добавление новости, просмотр новости, удаление новости) будет тестироваться аутентифицирован ли пользователь, который выполняет это действие (данные действия разрешены только авторизированным пользователям). Дополнительно для функции добавления проверяется корректность отправляемых данных, для функции просмотра – вывод всех новостей и вывод новостей по заданному фильтру, для функции удаления – удаление животного с несуществующим id, удаление животного не тем пользователем, который создал эту новость, и удаление правильным пользователем.

2.3 Стратегия интеграционного тестирования

Интеграционное тестирование — тестирование, при которой отдельные программные модули объединяются и тестируются в группе. В рамках проведения интеграционного тестирования тестируется взаимодействие модулей CreatePetFlow и NewsFlow. При создании животного (модуль CreatePetFlow), после добавления всех данных, пользователь нажимает на кнопку «сохранить». В итоге должно появиться сообщение об успешном добавлении, и должен произойти переход на страницу новостей (модуль NewsFlow).

Поэтому в рамках данного теста нужно проверить создание интента. Тестирование происходило прямо в android studio, был установлен breakpoint на строку кода, где создается intent.

2.4 Стратегия аттестационного тестирования

Аттестационное тестирование – это тестирование системы по функциональным требованиям. Оно используется для проверки, соответствует ли система заявленным требованиям.

Аттестационное тестирование подразумевает действия пользователя в интерфейсе, поэтому данное тестирование производилось в эмуляторе. Проводились действия, заявленные в тестах, полученные результаты сверялись с ожидаемыми результатами. Для тестирования использовался эмулятор Android Visual Device эмулирующий девайс Pixel 2 с API 27 и операционной системой Android 8.1.0.

Перед выполнением тестирования в базу данных добавляется информация о 2 пользователях и 6 собаках (каждый пользователь создает 3 объявления, одно из которых отмечается как "Найден"и одно имеет породу "Самоед")

2.5 Стратегия специального тестирования

Нагрузочное тестирование — подвид тестирования производительности, сбор показателей и определение производительности и времени отклика программно-технической системы или устройства в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе.

В рамках нагрузочного тестирования была проверена функция добавление новости, а именно часть, выполняющая загрузку дополнительных фотографий на сервер.

Осуществляется 4 теста в каждом из которых постепенно увеличивается количестово загружаемых фотографий начиная с 4 до максимального разрешенного количества - 15.

2.6 Условия начала, окончания и перехода между этапами тестирования

1. Этап тестирования считается завершенным, если пройдены все заявленные тесты и получены результаты (пройден / ошибка).

2. Можно начать следующий этап тестирования, только если предыдущий этап полностью завершен.

3. Тест считается успешно пройденным, если полученный результат совпадает с ожидаемым результатом.

4. Если полученный результат не совпадает с ожидаемым, следует проверить правильность теста. Если тест написан правильно, следует зафиксировать ошибку.

2.7 Условия возобновления и приостановки выполнения тестов

1. Тестирование может быть приостановлено только в случае ошибки, блокирующей выполнение тестирования.

2. Возобновление процедуры тестирования происходит после устранения ошибки, блокирующей выполнение тестирования.

3 План тестов

3.1 Модульное тестирование

Далее req - объект запроса, содержит информацию о теле запроса и заголовке запроса (тип передаваемого в теле контента, напрмер file, json и тд.; access_token для идентификации пользователя, посылающего запрос, уникальный для каждого пользователя); res - объект ответа.

1. **M-1** Цель теста – проверка создания новости неавторизированным пользователем (пустой идентификатор пользователя).
   Тип теста – негативный.
   Объект тестирования – функция authentication(req, res).
   Входные данные - пустое значение access_token по ключу Authorization.
   Ожидаемый результат - статус-код 401 Unauthorized.

2. **M-2** Цель теста – проверка создания новости неавторизированным пользователем (несуществующий идентификатор пользователя).
   Тип теста – негативный.
   Объект тестирования – функция authentication(req, res).
   Входные данные - неверное значение access_token по ключу Authorization.
   Ожидаемый результат - статус-код 401 Unauthorized.

3. **M-3** Цель теста – проверка создания новости пропавшего животного с несуществующей породой.
   Модель - pet.
   Тип теста – негативный.
   Объект тестирования – функция createObject(data = [], user).
   Входные данные - корректные значения name, color, age, description, gender, longitude, latitude, но значение breed не существует в БД.
   Ожидаемый результат - статус-код 200, res.body.success = false.

4. **M-4** Цель теста – проверка создания новости пропавшего животного с корректными данными.
   Модель - pet.
   Тип теста – общий.
   Объект тестирования – функция createObject(data = [], user).

Входные данные - корректные значения name, color, age, description, gender, longitude, latitude.

Ожидаемый результат - статус-код 200, res.body.success = false, res.body.data - объект.

5. **М-5** Цель теста – проверка просмотра новостей.

   Модель - pet.

   Тип теста – общий.

   Объект тестирования – функция findObjects(data = []).

   Входные данные - null.

   Ожидаемый результат - статус-код 200, res.body.success = true, res.body.data.pets - массив с 2 элементами.

6. **М-6** Цель теста – проверка просмотра новостей по фильтру.

   Модель - pet.

   Тип теста – общий.

   Объект тестирования – функция findObjects(data = []).

   Входные данные - age: “2”.

   Ожидаемый результат - статус-код 200, res.body.success = true, res.body.data.pets - массив с 1 элементом.

7. **М-7** Цель теста – проверка удаление животного с несуществующим id.

   Модель - pet.

   Тип теста – негативный.

   Объект тестирования – функция deleteObject(petId).

   Входные данные - несуществующий id животного.

   Ожидаемый результат - статус-код 200, res.body.success = false.

8. **М-8** Цель теста – проверка удаления животного не тем пользователем, который создал это животное.

   Модель - pet.

   Тип теста – негативный.

   Объект тестирования – функция deleteObject(petId).

   Входные данные - id животного не принадлежащего пользователю.

   Ожидаемый результат - статус-код 401 Forbidden.

9. **М-9** Цель теста – проверка удаления животного с корректными данными.

   Модель - pet.

Тип теста – общий.

Объект тестирования – функция deleteObject(petId).

Входные данные - id животного принадлежащего пользователю.

Ожидаемый результат - статус-код 200, res.body.success = true.

3.2 Интеграционное тестирование

1. **И-1** Цель теста – проверка вызова модуля новостей модулем создания объявления о пропаже животного.

   Тип теста – общий.

   Объект тестирования – модули новостей и создания объявления о пропаже животного.

   Входные данные - пользователь вводит текстовую информацию в поля name, age, description, выбирает breed и color из списка в полях автодополнения, выбирает gender в активити AddPetInfo, нажимает галочку, выбирает фотографию из галереи в активити AddPetImages, нажимает на галочку, устанавливает маркер на картев активити AddLocation, нажимает на галочку.

   Ожидаемый результат - в функции routeToNews() создается интент, открывается NewsActivity и появляется Toast с надписью “Pet added!”.

2. **И-2** Цель теста – проверка вызова модуля новостей модулем создания объявления о пропаже животного.

   Тип теста – общий.

   Объект тестирования – модули новостей и создания объявления о пропаже животного.

   Входные данные - пользователь вводит текстовую информацию в поля name, age, description, выбирает color из списка в поле автодополнения, вводит значение в поле breed, которого нет в списке автодополнения, выбирает gender в активити AddPetInfo, нажимает галочку, выбирает фотографию из галереи в активити AddPetImages, нажимает на галочку, устанавливает маркер на карте в активити AddLocation, нажимает на галочку.

   Ожидаемый результат - интент для перехода с CreatePetFlow на NewsFlow не создается, выводится Toast с сообщением "Please, choose existing breed!".

3.3 Аттестационное тестирование

1. **А-1** Цель теста – проверка возможности просмотра списка всех новостей.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - нажимается кнопка новостей.

   Ожидаемый результат - отображается список из 4 пропавших животных.

2. **А-2** Цель теста – проверка возможности просмотра списка животных по фильтру.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - в поле "Порода"выбирается "Самоед".

   Ожидаемый результат - отображается список из 2 отфильтрованных пропавших животных.

3. **А-3** Цель теста – проверка возможности просмотра и добавления закладок закладок пользователя.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - пользователь открывает объявление о пропавшем животном созданное другим пользователем и нажимает кнопку "Добавить в закладки". Открывает страницу новостей и нажимает на кнопку "Закладки".

   Ожидаемый результат - отображается список из 1 пропавшего животного.

4. **А-4** Цель теста – проверка возможности просмотра списка новостей, созданного пользователем.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - пользователь открывает страницу своих новостей.

   Ожидаемый результат - отображается список из 2 пропавших животных, созданных пользователем.

5. **А-5** Цель теста – проверка возможности просмотра архива новостей.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - пользователь открывает страницу своих новостей и нажимает

кнопку "Архив".

Ожидаемый результат - отображается список из 1 пропавшего животного.

6. **A-6** Цель теста – проверка возможности просмотра животных на карте.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - пользователь открывает страницу карты и уменьшает масштаб карты.

   Ожидаемый результат - при уменьшении карты показываются местоположение пропавших животных, которые находятся в данной зоне.

7. **A-7** Цель теста – проверка возможности входа в аккаунт.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - на экране аутентификации пользователь нажимает на кнопку "Sign in with Google если к его мобильному устройству не привязан аккаунт Google, ему предлагается аутентифицироваться в Google.

   Ожидаемый результат - пользователь проходит аутентификацию, открывается старница новостей.

8. **A-8** Цель теста – проверка возможности удаления информации о животном.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - нажимается кнопка delete на странице своего животного.

   Ожидаемый результат - выводится сообщение об удалении и открывается список своих пропавших животных.

9. **A-9** Цель теста – проверка возможности добавления пропавшего животного.

   Тип теста – общий.

   Объект тестирования – интерфейс приложения.

   Входные данные - вводится информация о пропавшем животном, добавляются основная и дополнительные фотографии, устанавливается местоположение.

   Ожидаемый результат - выводится сообщение об успешном добавлении и открывается страница новостей.

10. **A-10** Цель теста – проверка возможности написать другому пользователю.

    Тип теста – общий.

Объект тестирования – интерфейс приложения.

Входные данные - пользователь открывает объявление о пропаже животного созданное другим пользователем и нажимает на кнопку "Написать". Открывается чат, пользователь вводит сообщение в текстовое поле внизу жкрана и нажимает на кнопку "Отправить".

Ожидаемый результат - сообщение появляется в списке сообщений чата с правой стороны.

3.4 Специальное тестирование

1. **C-1** Цель теста – проверка производительности приложения при загрузке большого количества данных.

   Тип теста – общий.

   Объект тестирования – функция createPet(Pet pet).

   Входные данные - вводится информация о пропавшем животном, добавляются основная и 4 дополнительных фотографий, устанавливается местоположение.

   Ожидаемый результат - животное создается, сервис продолжает работать.

2. **C-2** Цель теста – проверка производительности приложения при загрузке большого количества данных.

   Тип теста – общий.

   Объект тестирования – функция createPet(Pet pet).

   Входные данные - вводится информация о пропавшем животном, добавляются основная и 8 дополнительных фотографий, устанавливается местоположение.

   Ожидаемый результат - животное создается, сервис продолжает работать.

3. **C-3** Цель теста – проверка производительности приложения при загрузке большого количества данных.

   Тип теста – общий.

   Объект тестирования – функция createPet(Pet pet).

   Входные данные - вводится информация о пропавшем животном, добавляются основная и 12 дополнительных фотографий, устанавливается местоположение.

   Ожидаемый результат - животное создается, сервис продолжает работать.

4. **C-4** Цель теста – проверка производительности приложения при загрузке большого количества данных.

Тип теста – общий.

Объект тестирования – функция createPet(Pet pet).

Входные данные - вводится информация о пропавшем животном, добавляются основная и 15 дополнительных фотографий, устанавливается местоположение.

Ожидаемый результат - животное создается, сервис продолжает работать.

3.5 Пример реализации теста

В качестве примера реализации тестов представлены тесты, проверяющие создание новости.

```
/*
 * CREATE PET
 */
describe('GET /pets/', () => {

    it('should return error when unauthenticated', (done) => {
        chai.request(app).post(`/pets/`).end((err, res) => {
            expect(res.status, res.status).to.be.equal(401);
            done();
        });
    });

    it('should return error when wrong access_token passed', (done) => {
        chai.request(app).post(`/pets/`).set('authorization', "invalid access_token").end((err, res) => {
            expect(res.status, res.status).to.be.equal(401);
            done();
        });
    });

    it('should return error when not existing breed is passed', (done) => {
        const data = {name: "Donut", breed: "Corgi", color: "Brown", age: "2", description: "Good boy", gender: "male", longitude: "1", latitude: "1"}
        chai.request(app).post(`/pets/`).set('authorization', `${USER_ACCESS_TOKEN}`).send(data).end((err, res) => {
            expect(res.status, "res.status").to.be.equal(200);
            expect(res.body.success, "res.body.success").to.be.false;
            done();
        });
    });

    it('should return success', (done) => {
        const data = { name: "Donut", breed: "Samoyed", color: "Brown", age: "2", description: "Good boy", gender: "male", longitude: "1", latitude: "1" }
        chai.request(app).post(`/pets/`).set('authorization', `${USER_ACCESS_TOKEN}`).send(data).end((err, res) => {
            expect(res.status, "res.status").to.be.equal(200);
            expect(res.body.success, "res.body.success").to.be.true;
            expect(res.body.data, "res.body.data").to.be.an('object');
            done();
        });
    });

})
```

Рис. 2: Пример реализации тестов

В 1 тесте проверяется создание животного незарегистрированным пользователем (передается пустой токен доступа пользователя), должен вернуться код 401.

Во 2 тесте передаваемый токен доступа пользователя неверный, должен вернуться код 401.

В 3 тесте проверяется создание животного несуществующей породы (все породы хранятся в отдельной таблице, и при создании животного берутся оттуда). Должен вернуться false.

4 тест - тест с верными данными, должен вернуться true.

Выполнение тестов:

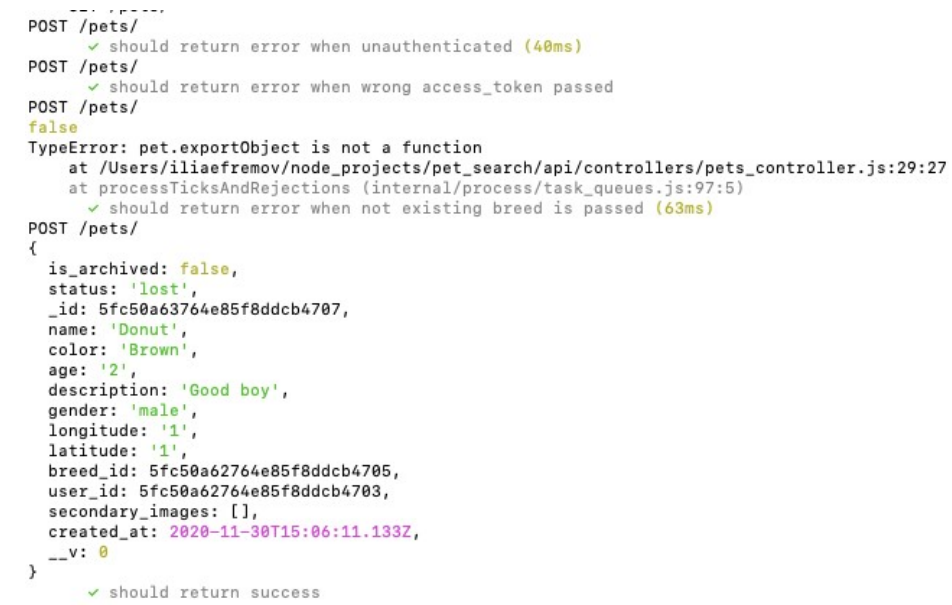

Рис. 3: Выполнение тестов

3.6 Оценка покрытия тестирования

В качестве метода оценивания покрытия тестирования было выбрано покрытие кода, которое рассчитывается по следующей формуле:

$$\text{Tcov} = (\text{Ltc}/\text{Lcode}) * 100\%, \text{ где}$$

Tcov - тестовое покрытие

Ltc - количество строк кода, покрытых тестами

Lcode - общее количество строк кода

$$\text{Tcov} = (531/\ 1023) * 100\% \approx 52\%$$

4 Журнал тестирования

4.1 Модульное тестирование

Номер теста	Дата	Тестировщик	Результат	Номер ошибки
1	26 ноября 2020	Жданович Е. С.	Пройден	—
2	26 ноября 2020	Жданович Е. С.	Пройден	—
3	26 ноября 2020	Жданович Е. С.	Пройден	—
4	26 ноября 2020	Жданович Е. С.	Пройден	—
5	26 ноября 2020	Жданович Е. С.	Пройден	—
6	26 ноября 2020	Жданович Е. С.	Ошибка	№1
7	27 ноября 2020	Жданович Е. С.	Пройден	—
8	27 ноября 2020	Жданович Е. С.	Пройден	—
9	27 ноября 2020	Жданович Е. С.	Пройден	—

4.2 Интеграционное тестирование

Номер теста	Дата	Тестировщик	Результат	Номер ошибки
1	28 ноября 2020	Жданович Е. С.	Пройден	—
2	10 декабря 2020	Жданович Е. С.	Пройден	—

4.3 Аттестационное тестирование

Номер теста	Дата	Тестировщик	Результат	Номер ошибки
1	25 ноября 2020	Жданович Е. С.	Пройден	—
2	25 ноября 2020	Жданович Е. С.	Ошибка	№2
3	25 ноября 2020	Жданович Е. С.	Пройден	—
4	25 ноября 2020	Жданович Е. С.	Пройден	—
5	14 декабря 2020	Жданович Е. С.	Пройден	—
6	14 декабря 2020	Жданович Е. С.	Пройден	—
7	14 декабря 2020	Жданович Е. С.	Пройден	—
8	15 декабря 2020	Жданович Е. С.	Пройден	—
9	15 декабря 2020	Жданович Е. С.	Пройден	—
10	15 декабря 2020	Жданович Е. С.	Пройден	—

4.4 Специальное тестирование

Номер теста	Дата	Тестировщик	Результат	Номер ошибки
1	10 декабря 2020	Жданович Е. С.	Пройден	—
2	15 декабря 2020	Жданович Е. С.	Пройден	—
3	15 декабря 2020	Жданович Е. С.	Пройден	—
4	15 декабря 2020	Жданович Е. С.	Пройден	—

5 Журнал найденных ошибок

1. **Ошибка №1.**

   Тест: M-6.

   Входные данные: age: “2”.

   Ожидаемый результат - статус-код 200, res.body.success = true, res.body.data.pets - массив с 1 элементом.

   Фактический результат: res.body.data.pets - массив с 2 элементами.

   Состояние: исправлено.

2. **Ошибка №2.**

   Тест: A-2.

   Входные данные: ввод параметров фильтрации на странице новостей.

   Ожидаемый результат: отображается список отфильтрованных пропавших животных.

   Фактический результат: отобразился список всех пропавших животных.

   Состояние: исправлено.

6 Результаты

Во время выполнения тестирования было найдено несколько ошибок. В результате ошибки были исправлены. Также при выполнении аттестационного тестирования были выявлены недостатки пользовательского интерфейса, не существенные, но способные доставить неудобство пользователю при работе с приложением.

Приложение было протестировано не полностью, а только его часть. Далее я планирую протестировать оставшиеся части, так как в будущем планируется выпуск этого приложения.

Тестирование можно считать пройденным успешно.