**Автотесты**

Проект «Гео Таргет» представляет собой приложение для таргетированной рекламы в Telegram с учетом геолокации пользователей. Эффективность и надежность такого приложения напрямую зависят от качества его технической реализации, что делает автотестирование критически важным компонентом разработки.

Цели автотестирования:

* Обеспечение функциональной корректности всех API;
* Подтверждение надежности и безопасности обработки данных;
* Гарантирование быстрого определения и устранения возникающих проблем.

Используемые технологии:

* Python 3.11.4: Основной язык программирования;
* pytest 7.4.0: Фреймворк для организации и запуска тестов;
* pydantic 2.3.0: Используется для валидации данных и моделирования бизнес-объектов;
* httpx 0.24.1: Для выполнения асинхронных HTTP запросов к API.

Написание тестов

Тесты разрабатываются с учетом следующих ключевых принципов:

* Идейность: Каждый тест фокусируется на одной функциональной идее;
* Атомарность: Тесты изолированы и проверяют только одну функцию или состояние;
* Независимость: Возможность запуска тестов в любом порядке без предварительной настройки состояния системы;
* Гибкость: Легкая адаптация к изменениям в коде и тестовой среде;

Требования к структуре наших тестов. Каждый тест имеет обязательные и необязательные атрибуты (зависит от логики самого теста):

1. Подготовка данных: Настройка необходимых данных и состояний перед выполнением теста;
2. Отправка запроса;
3. Проверка кода ответа (сравнивается ожидаемый и пришедший код);
4. Проверка схемы ответа (проверяется структура тела ответа и типы полей);
5. Проверка тела ответа и специфической логики;
6. Очистка: Удаление или возврат тестовой среды к исходному состоянию, если это необходимо.

1 и 2 пункты исполняем обязательно т. к. любой запрос дает нам ожидаемый код ответа. Третий пункт будем проверять, если у ответа есть тело (json). 4 пункт подразумевает проверку корректности значений полей в теле ответа, а также специфической логики, которая соответствует идеи нашего теста. Этот пункт будет обязательным, если у ответа есть тело.

Шаблон с принципами и действиями, которыми будем руководствоваться при написании чек-листов.

GET

Цель: Проверка правильности получения данных, связанных с геолокацией для рекламных кампаний.

|  |  |
| --- | --- |
| Метод | Пункты, которые необходимо реализовать |
| GET | 1) Проверка корректности получения объекта: - В тестах заранее заготовлен файл с данными для проверки на эталонность без айдишников (файл одинаков при каждом запуске), в базе заранее заготовлен объект, который никогда не изменяется; - Делаем get запрос с нужным параметром; - Делаем полное сравнение ответа с эталонным файлом.  2) Проверка специфической логики. |

Распишем чек-листы, распределив их на позитивные и негативные.

|  |  |
| --- | --- |
| Позитивные | Негативные |
| Запрос с параметрами по-умолчанию;  Запрос с 1 сущ. айдишником;  Запрос с 2 сущ. айдишниками. | Запрос с несущ. айдишником;  Запрос с невалидным айдишником. |

POST

Цель: Проверка функций создания новых рекламных кампаний с привязкой к геолокации.

|  |  |
| --- | --- |
| Метод | Пункты, которые необходимо реализовать |
| POST | 1) Проверка корректности сохранения объекта на сервере: - Создаем объект; - Запоминаем отправляемый объект; - Отправляем объект; - Убеждаемся, что объект в ответе корректен; - Убеждаемся, что в базе этот объект корректно сохранен.  2) Проверка специфической логики. |

|  |  |
| --- | --- |
| Позитивные | Негативные |
| Запись пустого тела в базу;  Запись name и data с полями всех типов в базу. | Отправка невалидного json. |

PUT

Цель: Тестирование обновления информации о рекламных кампаниях.

|  |  |
| --- | --- |
| Метод | Пункты, которые необходимо реализовать |
| PUT | 1) Проверка корректности обновления объекта в базе: - Получаем id обновляемого объекта; - Создаем объект с другими значениями; - Запоминаем состояние объекта до обновления его в базе; - Обновляем в базе объект; - Убеждаемся, что объект в ответе корректно обновлен; - Убеждаемся, что в базе этот объект корректно сохранен.  2) Проверка специфической логики. |

|  |  |
| --- | --- |
| Позитивные | Негативные |
| Обновление name и data с полями всех типов;  Обновление на пустой. | Обновление несущ. объекта  на невалидный объект. |

DELETE

Цель: Проверка удаления данных о кампаниях.

|  |  |
| --- | --- |
| Метод | Пункты, которые необходимо реализовать |
| DELETE | 1) Проверка удаления объекта с сервера: - Получаем id удаляемого объекта; - Удаляем его; - Убеждаемся, что он удален (404 ошибка).  2) Проверка специфической логики. |

|  |  |
| --- | --- |
| Позитивные | Негативные |
| Удаление существующего объекта. | Удаление несуществующего объекта. |

Дополнительные возможные тесты:

* Тесты производительности: Проверка времени ответа приложения при высокой нагрузке;
* Тесты безопасности: Проверка устойчивости к атакам, таким как SQL-инъекции и переполнение буфера;
* Интеграционные тесты: Проверка взаимодействия с внешними сервисами, такими как сервисы карт и геолокации.

Каждый тест соблюдает 4 сформированных требования. Тесты атомарны и независимы, а значит мы можем без проблем изменить 1 тест, не нарушая логику другого теста. Тесты относительно гибки.

Тщательное и всестороннее тестирование критически важно для успеха приложения «Гео Таргет». Эта документация должна служить основой для создания и поддержки качественной тестовой среды, обеспечивающей высокий уровень надёжности и производительности приложения. Внедрение автоматизированных тестов поможет выявлять потенциальные проблемы на ранних этапах разработки и сократить время на дебаг и рефакторинг.

Процесс управления тестированием:

* Планирование: Определение целей тестирования, выбор инструментов и формирование тестовой стратегии;
* Разработка тестов: Написание тест-кейсов по утвержденной спецификации Разработка тестов включает создание тестовых данных и сценариев;
* Выполнение тестов: Запуск тестов в автоматизированной среде. Регулярное обновление и повторение тестов для проверки стабильности и производительности;
* Анализ результатов: Оценка результатов тестирования, выявление дефектов и анализ их причин;
* Оптимизация и обслуживание: Внесение изменений в тесты в ответ на обновления в приложении и изменения в требованиях.

Инструменты и инфраструктура

Для поддержки тестирования потребуется настроить специализированное тестовое окружение, которое включает:

* Тестовые серверы для имитации реальных условий работы приложения.
* Системы контроля версий для управления версиями тестов и исходного кода.
* Системы непрерывной интеграции/развертывания (CI/CD), которые автоматически запускают тесты при каждом обновлении кода.
* Мониторинг и логирование для сбора данных о производительности приложения во время тестирования.

Сценарии использования и тестовые кейсы

Для каждого метода API (GET, POST, PUT, DELETE) следует разработать набор сценариев, охватывающих как стандартное использование, так и потенциальные исключительные ситуации. Тестовые кейсы должны включать:

* Позитивные тесты для проверки ожидаемого поведения функций при корректных данных;
* Негативные тесты для проверки устойчивости системы к ошибочным или атакующим вводам;
* Граничные тесты для проверки системы на пределах допустимых значений данных.

Примеры конкретных тестов:

1. Тестирование геолокационной фильтрации: Проверка, что рекламные объявления корректно отображаются для пользователей в определённой географической зоне.
2. Тестирование реакции на изменение геолокации: Симуляция изменения местоположения пользователя и проверка соответствующего изменения в предлагаемой рекламе.

Выводы

Подготовленная документация по автотестированию станет основой для построения надёжной и эффективной системы тестирования в проекте «Гео Таргет». Продуманное и систематическое тестирование позволит не только обеспечить высокое качество продукта, но и значительно сократить время и ресурсы, затрачиваемые на поиск и устранение ошибок в будущем.