**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ITMO University**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 2**

**По дисциплине** Тестирование программного обеспечения

**Тема работы** Интеграционное тестирование

**Обучающийся** Алексеев Тимофей Юрьевич

**Факультет** Факультет инфокоммуникационных технологий

**Группа** К3321

**Направление подготовки** 11.03.02 Инфокоммуникационные технологии и системы связи

**Образовательная программа** Программирование в инфокоммуникационных системах

**Обучающийся** 23.11.2024  Алексеев Т.Ю.

(дата) (подпись) (Ф.И.О.)

**Руководитель**  Кочубеев Н.С.

(дата) (подпись) (Ф.И.О.)

Санкт-Петербург  
2024 г.

# Цель

Научиться разрабатывать и применять интеграционные тесты для проверки взаимодействия между компонентами в существующем проекте.

# Задачи

1. Выбрать открытый репозиторий на GitHub;
2. Провести анализ взаимодействий между компонентами;
3. Написать 5 интеграционных тестов.

# Ход работы

**Выбор репозитория**

В данной работе необходимо выбрать любой открытый проект на GitHub для дальнейшей работы. Мною был выбран мой личный пет-проект, который делался в качестве итогового по предмету «Программирование».

Программа представляет из себя удобной приложение с графическим интерфейсом для учета автопарка логистической компании. В приложении можно добавлять и удалять машины, смотреть их список и подбирать их с помощью специальных фильтров, учитывая грузоподъемность и занятость машины.

**Анализ взаимодействий между модулями**

В данном проекте есть всего 3 модуля, которые взаимодействуют друг с другом и выводят информацию в главном файле main.py.

В файле Transports.py создается класс транспорта, по образу которого далее создаются все машины, чтобы в дальнейшем удобнее работать с ними. В файле Baseofcars.py содержится класс CarBase, который содержит в себе все методы автопарка, а именно различные сортировки, удаления, добавления машин и тд. В файле DB.py содержится класс DataBase, который содержит в себе методы по управлению базой данных Transports.db (sqlite3).

*Ключевыми точками интеграции в данном проекте являются:*

1. Передача измененных данных в базу данных (Transport.db) из файла DB.py;
2. Передача сохраненных данных из базы данных (Transport.db) в файл main.py для отображения;
3. Передача отсортированных данных из файла Baseofcars.py в файл main.py для отображения.

*Критические части системы:*

1. Взаимодействие приложения с базой данных путем SQL-запросов;
2. Взаимодействие модулей DB.py, Baseofcars.py, Transports.py и файла main.py, где вызываются функции объектов из модулей.

*Важные сценарии взаимодействия:*

1. Добавление нового транспортного средства в БД;
2. Удаление существующего транспортного средства в БД;
3. Обновление занятости транспортного средства в БД;
4. Выведение списка транспортных средств с определенной грузоподъемностью;
5. Выведение полного списка транспортных средств;
6. Выведение списка занятых транспортных средств;
7. Выведение списка свободных транспортных средств;
8. Выведение списка отсортированных по грузоподъемности транспортных средств.

**Написание тестов**

Тесты были написаны на языке Python с помощью библиотеки unittest.

В первом тесте проверка состояла в добавлении новой машины. Для этого мы добавляли машину в БД, после чего вызывали все строки из БД и проверяли, есть ли добавленная запись в них.

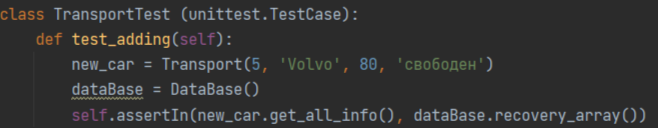


Рисунок 1 – Тест на добавление транспортного средства

Во втором тесте проходит проверка на удаление транспортного средства, существующего в базе данных. Логика передачи информации такая же, как и в первом тесте.

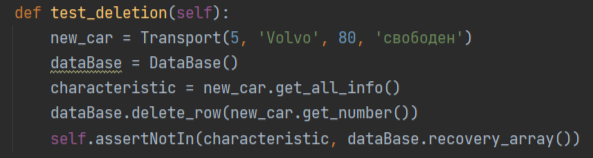


Рисунок 2 – Тест на удаление транспортного средства

В третьем тесте проверялась взаимодействие базы данных и файла Baseofcars.py, который работает с сортировкой и фильтрацией машин. Здесь проверялась функция, выводящая машины, уровень грузоподъёмности которых не ниже введенного пользователем. Для этого добавлялась машина с максимальной грузоподъемностью и проверялось, что при запросе в финальном списке выводится только она.

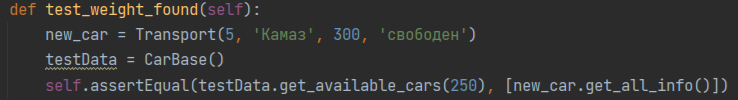


Рисунок 3 – Тест на фильтрацию транспорта по грузоподъемности

В четвертом тесте проверялось бронирование транспортного средства, то есть обновление статуса машины в БД. Для этого необходимо задействовать сразу все 3 модуля. Сначала создавалась свободная машина в модуле Transport.py, после чего она бронировалась через Baseofcars.py и записывалась в БД с помощью DB.py.

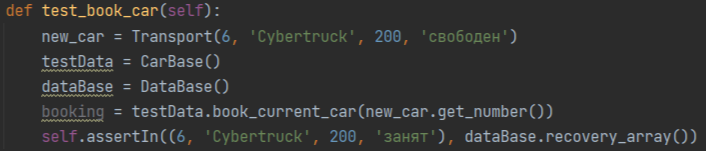


Рисунок 4 – Тест на бронирование транспортного средства

В пятом тесте проверялся вывод свободных машин, а именно взаимодействие базы данных и модуля Baseofcars.py. Для этого вызывается метод из класса BaseCar, который выводит список свободных транспортных средств. После чего он сравнивается с вручную отфильтрованным списком из базы данных.

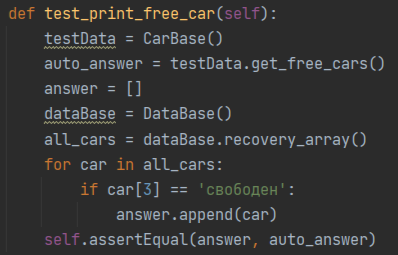


Рисунок 5 – Тест на вывод списка свободных машин

**Результаты тестирования**

Все тесты прошли успешно. Это показало нам идеальную работу с базой данных и ее синхронизацию. Также, вызов функций из других модулей в модулях не вызывает проблем. Ни одной ошибки найдено не было.

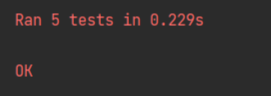


Рисунок 6 – Результаты тестирования

Единственное место, где могут возникать ошибки, — это некорректный пользовательский ввод. Но данная область относится к unit-тестированию и в данной работе не затрагивается.

**Вывод**

В ходе выполнения лабораторной работы цель была достигнута. Научились разрабатывать и применять интеграционные тесты для проверки взаимодействия между компонентами в существующем проекте.

Был выбран личный пет-проект, был проведен анализ его компонентов и взаимодействий между ними: ключевые точки интеграции, критические части системы и важные сценарии взаимодействия. После чего были написаны 5 интеграционных тестов, которые и проверили данное взаимодействие. Ошибок выявлено не было.