Содержание

[Содержание 2](#_Toc343592082)

[Анализ предметной области 3](#_Toc343592083)

[Постановка задачи 5](#_Toc343592084)

[Требования к создаваемому программному средству 5](#_Toc343592085)

[Входные/выходные данные, их спецификация 5](#_Toc343592086)

[Требования к интерфейсу 6](#_Toc343592087)

[Используемая платформа 6](#_Toc343592088)

[Состав команды: 10](#_Toc343592089)

[Схема базы данных 11](#_Toc343592090)

[UseCase диаграмма 12](#_Toc343592091)

[Диаграмма классов 13](#_Toc343592092)

[Пользовательский интерфейс 14](#_Toc343592093)

[Тестирование 15](#_Toc343592094)

[Выводы 18](#_Toc343592095)

[Системные требования 19](#_Toc343592096)

[Руководство пользователя 20](#_Toc343592097)

Анализ предметной области

Распознавание автомобильных номеров. Данной задачей занимаются как специализированные организации, так и студенты различных вузов по всему миру. Отличие таких программ: во-первых, организации пишут код высокого уровня, процент распознавания номеров такими программами стремится к 100%, а во-вторых, такой код не является открытым.

Одной из первых российских компаний, занимающихся распознаванием номеров, является ООО “МегаПиксел”, их продукт называется “MegaCar”, первая версия которого вышла в 1995 году. Сейчас же рынок предлагает десятки таких коммерческих продуктов. Даже уже существующие работающие системы постоянно совершенствуются, так как существует необходимость распознавать номера в различных погодных условиях (особенно эта проблема касается распознавания на территории России), на различной скорости, под различными углами. Совершенствуются методы распознавания. Так сначала распознавали только фронтальные изображения, потом стали распознавать под углом. Сейчас же существуют методики, которые позволяют распознать номера (в том числе и любой текст) в произвольно положении. Отдельными проблемами являются сложный фон изображений и помехи.

В связи с особой сложностью распознавания текста в произвольном положении пути решения данной проблемы развивались постепенно, были разработаны три метода:

1. texture-based;
2. region-based;
3. hybrid.

Данное программное средство позволит выделять из фотографии и распознавать номер автомобильного средства. В целом данное программное средство находит или может найти спрос в полиции, магазинах, отелях, парковках (особенно «умным», которые ведут машину к нужному месту), пропускным системам и т.д. Такое программное средство уже реализовано, но код проекта закрыт и как правило эти решения интегрированы с аппаратной частью. Данный же проект основывается на открытом коде, возможно, дальнейшем данное средство будет реализовано под мобильные операционные системы, такие как Android и iOS.

Постановка задачи

Создать веб-сайт, который:

* Позволяет загрузить изображение на сервер
* Распознать номер машины по изображению

Возможно, в будущем создать мобильное приложение под Android и iOS

Приложение будет использовать OpenCV, который уже нормально портирован на Android и iOS. Что в дальнейшем существенно облегчит задачу расширения проекта.

Требования к создаваемому программному средству

Входные/выходные данные, их спецификация

Входные данные: фотография с фронтально расположенной машиной

Выходные данные: выделенная на фотографии область с номером машины, сам номер машины

Требования к интерфейсу

* Дружественный, понятный пользователю
* Позволяющий загрузить фотографию
* Четко выводящий распознанный номер
* Возможность отослать разработчикам неправильно распознанный, или же совсем не распознанный номер для последующего улучшения библиотеки

Используемая платформа

Язык программирования: Python 2.7

Гибкий, динамически типизируемый язык, с низким порогом вхождения. Позволяет быстро разрабатывать прототипы, в отличие от C++ и Java. Удобен для написания нейронных сетей и обработки изображений, и более распространен чем Matlab/Ocatve.

Среда разработки: PyСharm 2.6, IDLE, vim

Операционная система: Windows, Linux (Ubuntu). В дальнейшем планируется портировать на платформы iOS и Android. Потенциально возможено использование на любых платформах которые поддерживают OpenCV и Python

Библиотеки: OpenCV. Низкоуровневая библиотека для работы с машинным зрением. Предоставляет реализации огромного числа алгоритмов для работы с изображениями и видео-потоком. Может использоваться в большинстве современных языков, включая C/C++, Java, C#, Matlab, Python, и т.д.

Framework Django : удобный фреймворк для создания веб-сайтов на языке python, что позволяет нам без проблем использовать алгоритм написанный на этом же языке.

Обеспечивает быстрое создание веб-сайта любой сложности

СУБД : MySQL - доступная, и распространенная СУБД

**Планирование работ**

Локализация автомобильного номера

На вход подается фотография, на выходе получаем список четырёхугольников - регионов, которые подходят под критерии номеров.

Локализация сводится к использованию алгоритма SWT со специальными фильтрами.

Распознавание символов автомобильного номера

На вход подаётся вырезанный из картинки номер. На выходе должна быть последовательность символов. Эта задача делится на три этапа:  Препроцессинг, выделение отдельных символов, распознавание символов.

Препроцессинг.

Во-первых, номер может быть повернут. Во-вторых, он может быть неодинаково освещён, в третьих, он может быть нечётким. На этом этапе в зависимости от рассчитанных характеристик изображения номер надо автоматически развернуть, нормализовать освещение и применить фильтры, делающие изображение более чётким.

Выделение отдельных символов.

Необходимо провести кластеризацию картинки и выделить регионы, соответствующие отдельным символам.

Распознавание символов.

Для распознавания используется свободный OCR движок Tesseract.

В планах переход на свой классификатор, в виде натренированной нейронной сети с обратным распространением ошибки.

Для выделения отдельных символов используется алггоритм Stroke Width Transform разработанный в MS Research.

Основная идея алгоритма состоит в поиске областей с “штрихами” равной ширины, которые как правило соответствуют печатным буквам. Затем происходит фильтрация полученных областей.

Подробно:

1) Получение контуров на изображении (используется детектор границ Canny из OpenCV)

2) Получаем градиент с помощью оператора Собеля (так же реализация из OpenCV)

3) Для каждой точки контура повторяем - идем от точки в сторону ее градиента, пока не окажемся на контуре опять, если градиент в данной точке противоположен градиенту начальной, то добавляем отрезок в список отрезков. Если мы оказались на границе изображения, то прекращаем поиск для этой точки

4) Создаем карту длин лучей, по принципу - каждой точке присваивается длинна наименьшего луча проходящего через данную точку.

5) Ищем компоненты связности на полученной карте

6) Фильтруем компоненты с помощью набора правил

7) Выбираем из оставшихся компонент пары компонент близких по параметрам (таких как высота/ширина)

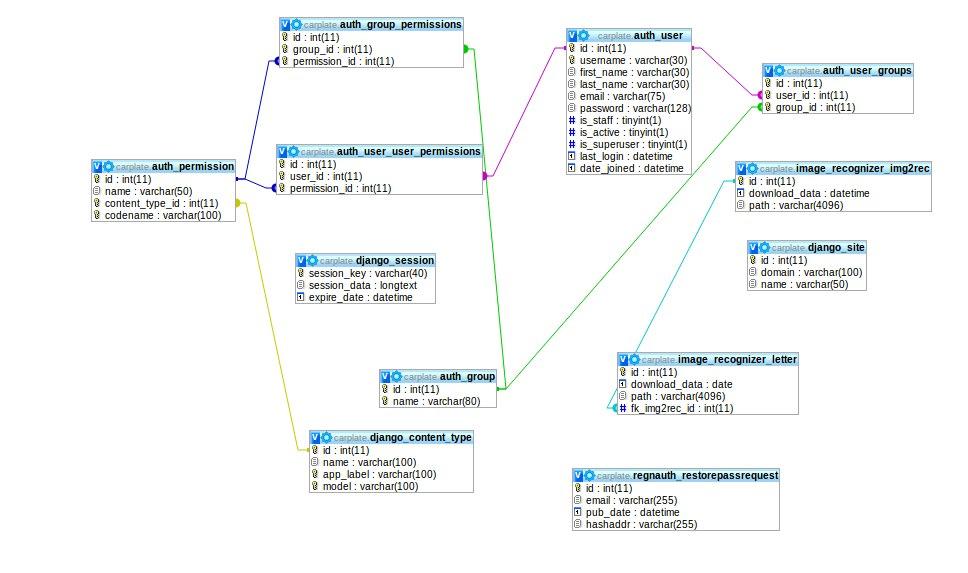
Для создания собственно веб-сайта используется фреймворк Django. С полностью реализованным паттерном MTV (MVC), который является неотъемлимой частью Django, получили гибкое приложение с легко изменяемым интерфейсом без ущерба логике проекта.

Состав команды:

Климов Кирилл (teamlead) – реализация SWT, небольшая часть работы с сокет-сервером и django, тестирование

Манущева Алина – Основная часть работы над сокет-сервером, веб-сайт на Django, документация

Схема базы данных



UseCase диаграмма

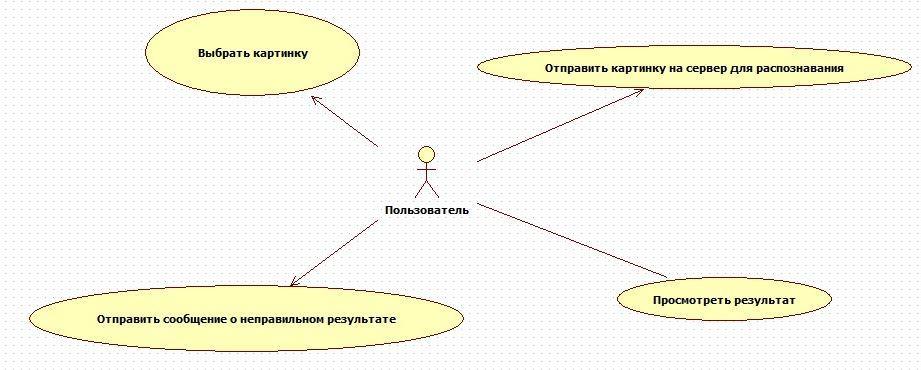
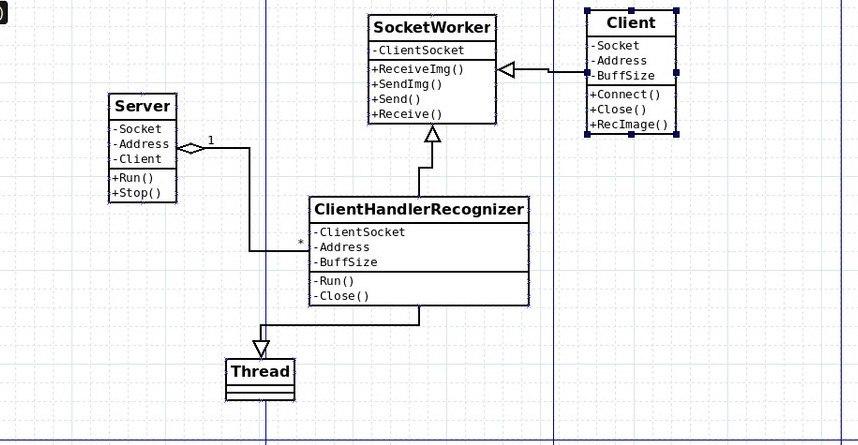
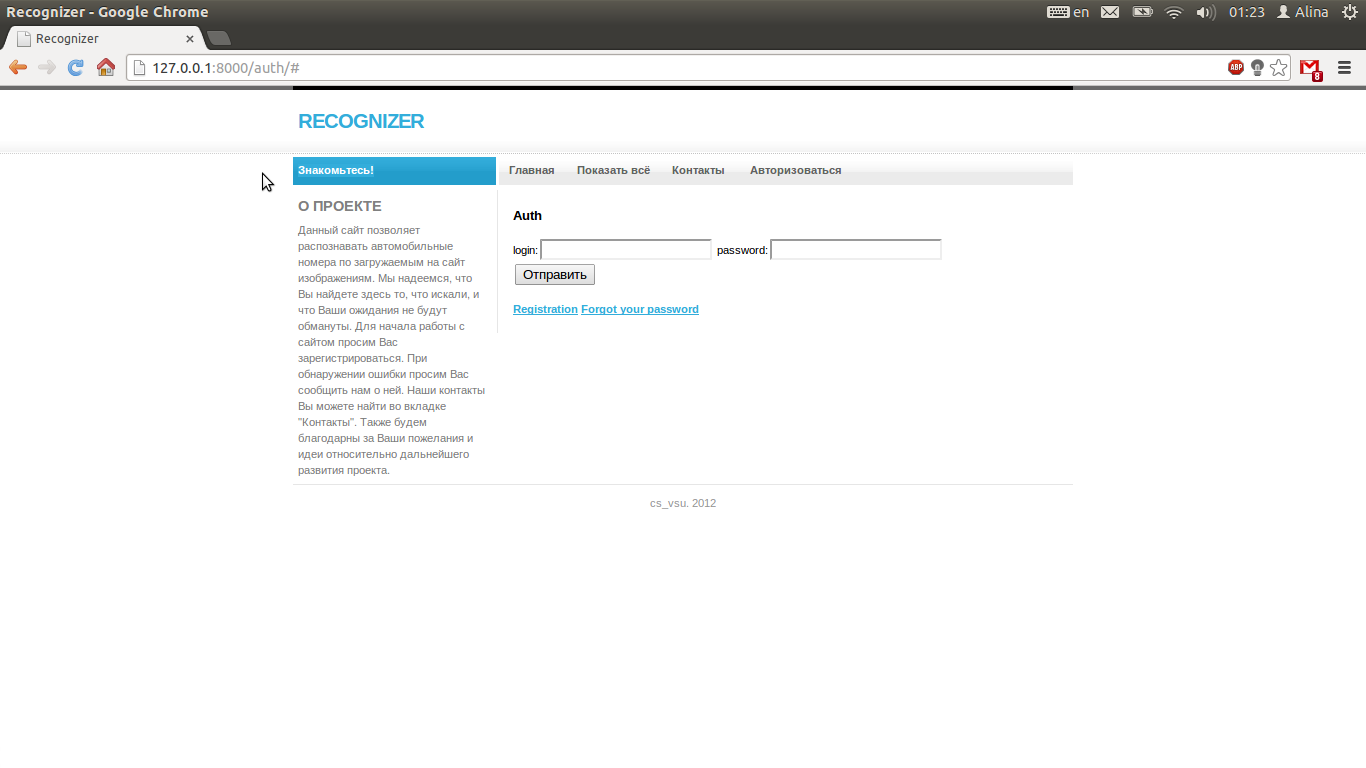
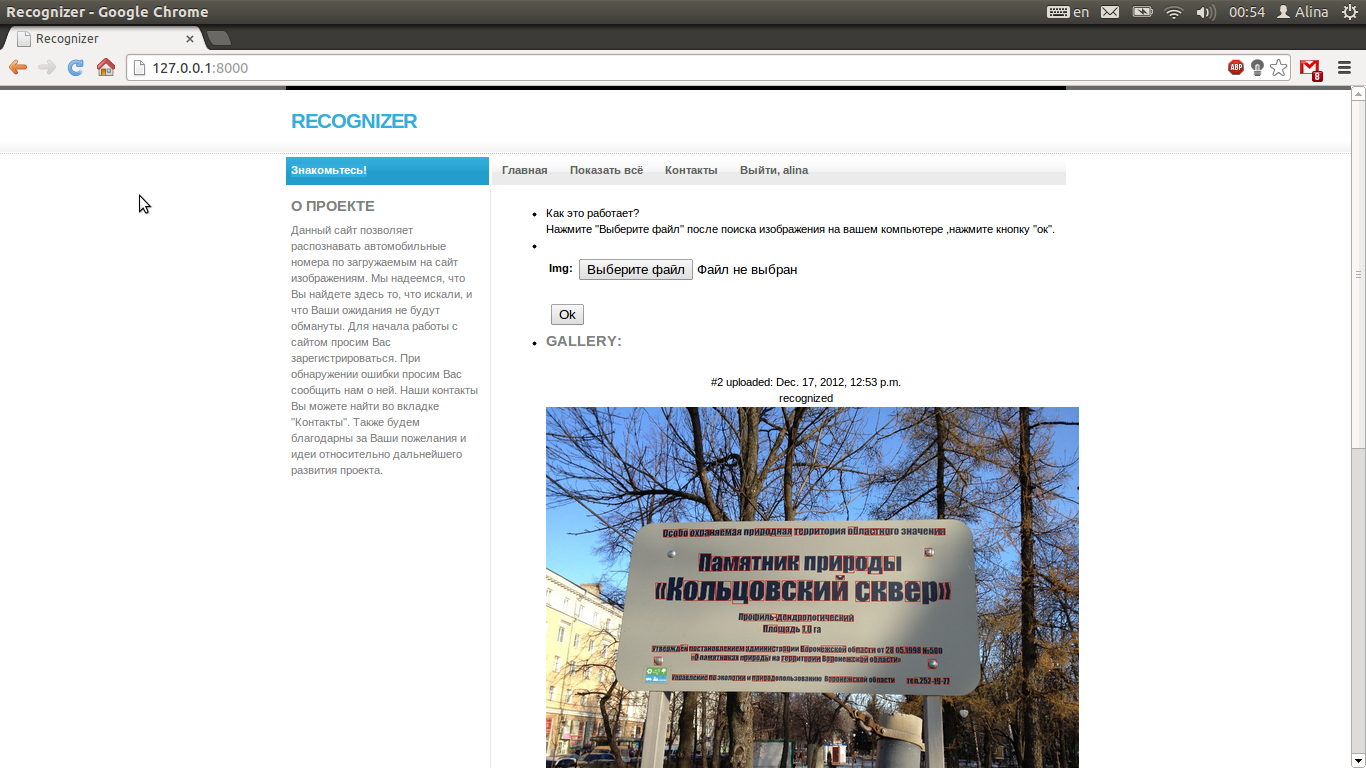


Диаграмма классов



Пользовательский интерфейс

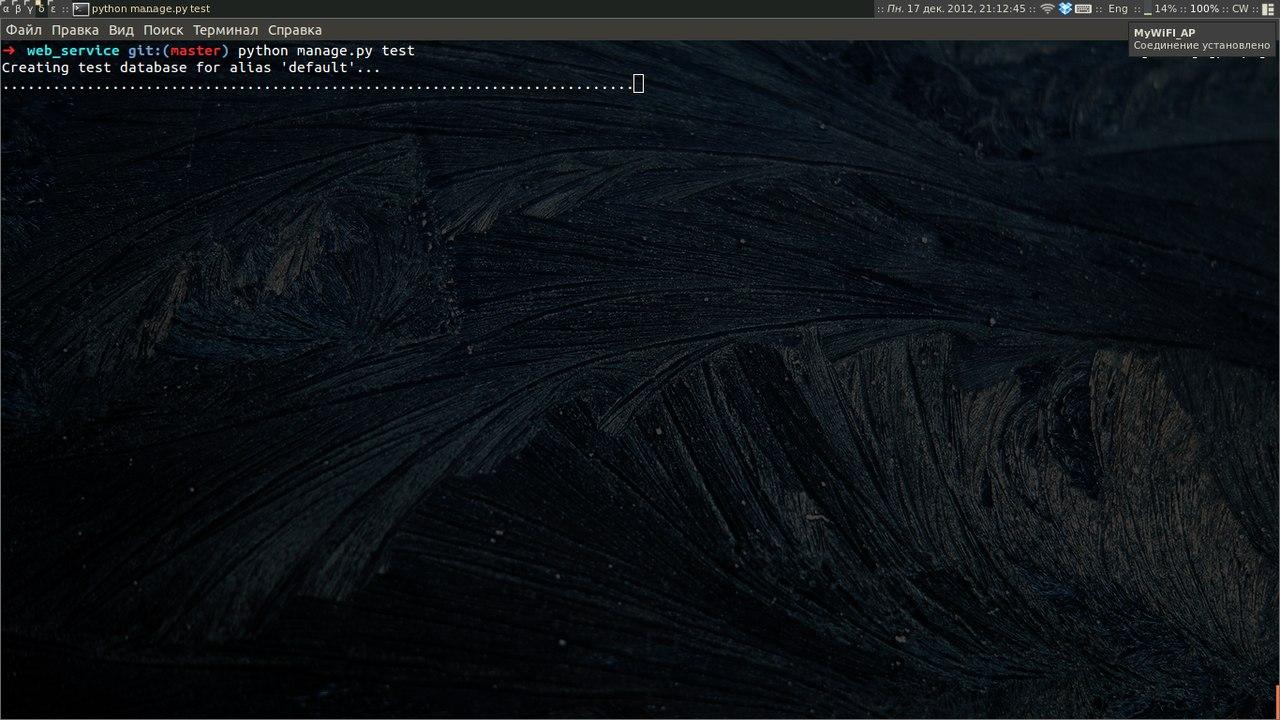


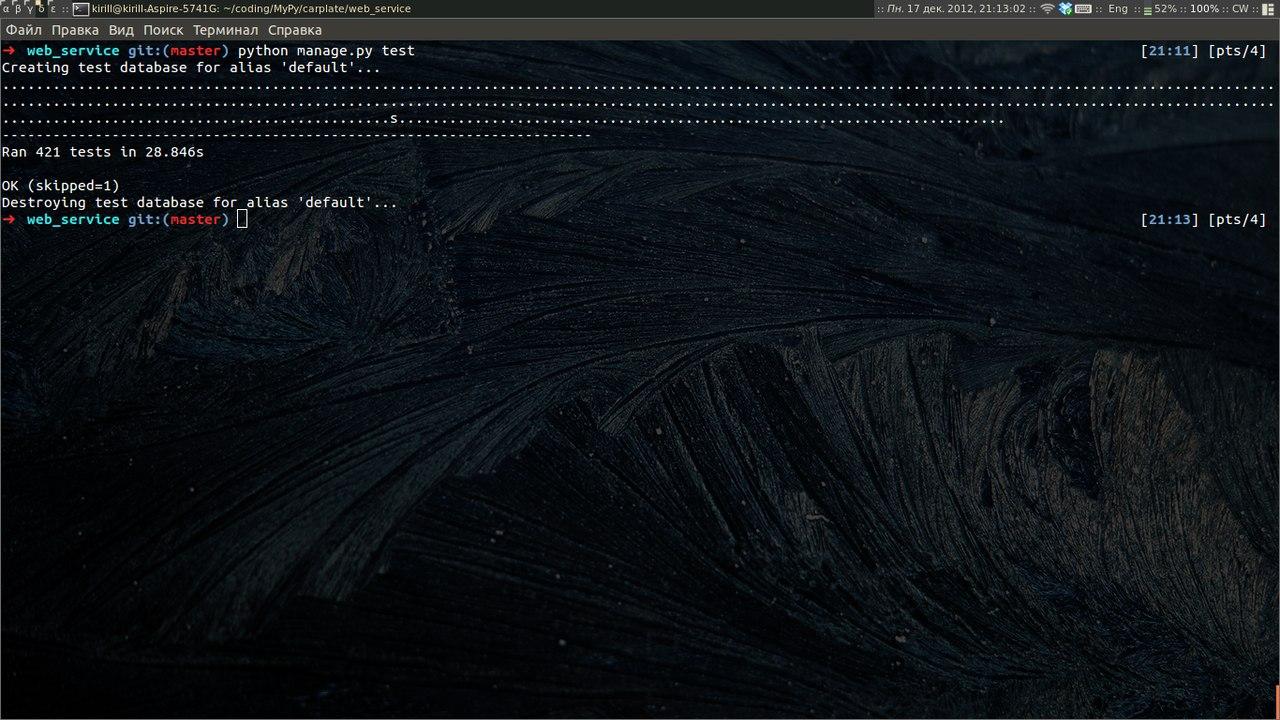




Тестирование

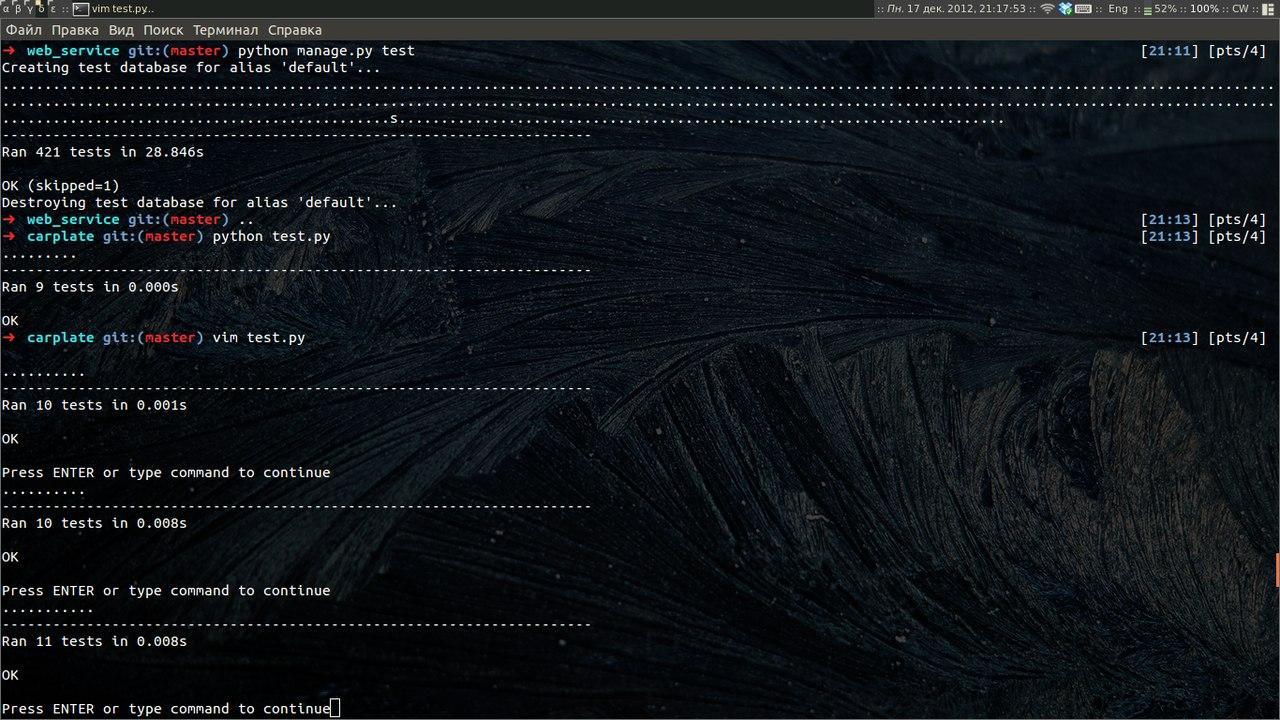
**1.** Встроенное тестирование Django

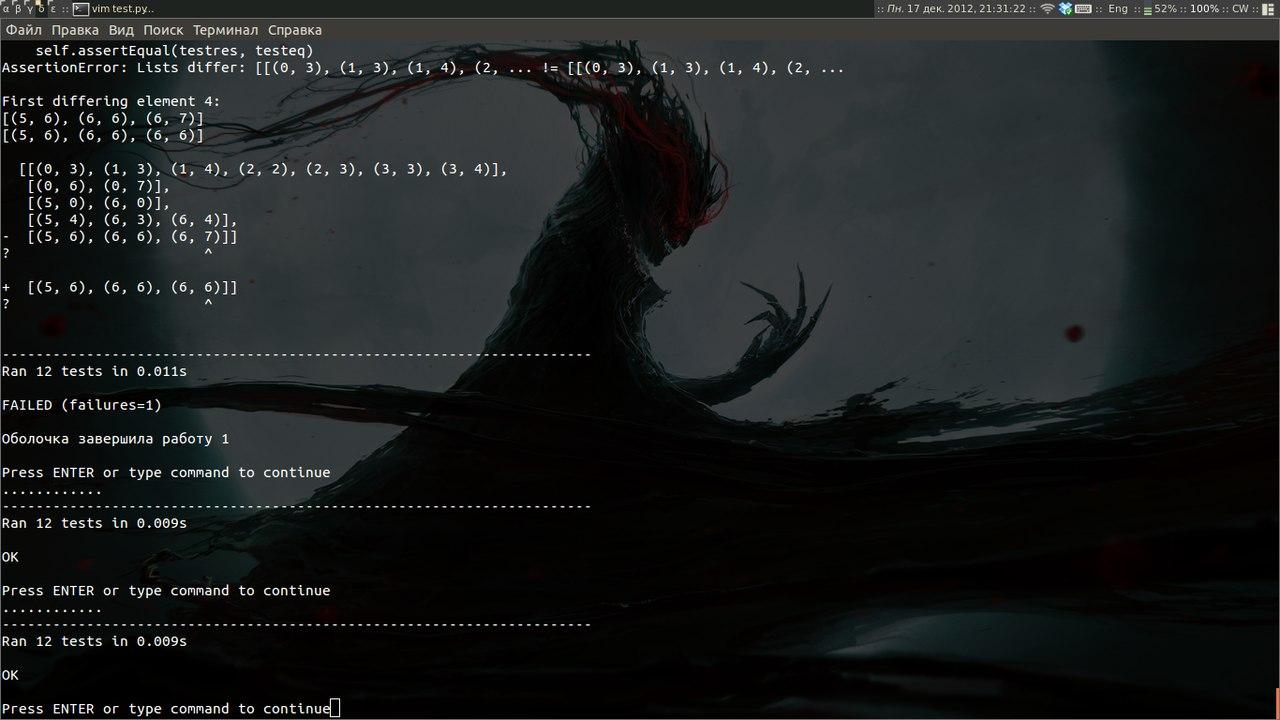


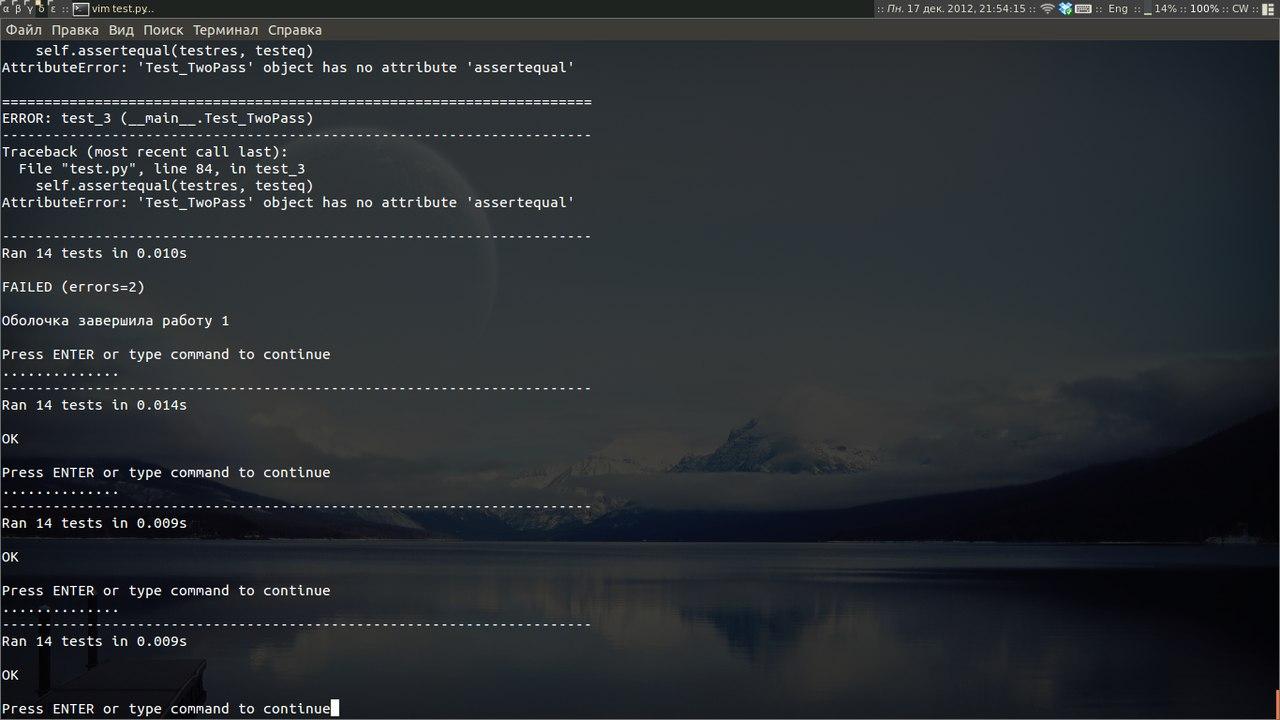


**2**. Тестирование SWT

1. Проверка математических функций поиска направления двух градиентов
2. Проверка функции, ищущей расстояние между двумя компонентами
3. Тестирование нескольких типовых примеров в алгоритме кластеризации компонент







Тестирование позволяет проверить программу на правильность работы, проверить сложные вычисления, работу при больших нагрузках.

Были протестированы основные математические функции. Тестирование показало, что ошибок не обнаружено.

Выводы

Готовое приложение соответствует нашим ожиданиям и начальному описанию. Приложение работает корректно, но хотелось бы увеличить производительность, сократить время ожидания результата.

Так же есть куда развивать проект, т.к. не реализованы полностью все желаемые дополнительные функции.

Системные требования

Для запуска сервиса

* ОС: Ubuntu 11.10 и выше (проект тестировался на версии 12.10)
* Интерпретатор Python 2.7
* СУБД MySQL
* Браузер

Библиотеки:

1. NumPy
2. OpenCV
3. Parallel Python
4. Фреймворк Django
5. MySQLdb

Руководство пользователя

Для локального запуска проекта, необходимо, чтобы компьютер удовлетворял системным требования, приведенным выше, т.е. были установлены соответсвующие операционная система, интерпретатор и сервер базы данных, а так же все необходимые библиотеки, фреймворк и коннектор базы данных, а так же сама база данных хранилась на localhost.

Последовательность действий:

1. Запуск MySQL сервера
2. Запуск сервера проекта :

cd carplate

python service-server.py 20000

1. Запуск сервера Django :

cd carplate/web-service/

python manage.py runserver <номер свободного порта[1]>

1. Открыть браузер, в адресной строке ввести http://127.0.0.1:номер порта[1]/