Оглавление

TODO

Вступление

В современном мире, когда технологии развиваются с невероятной скоростью, многие сферы жизни человека претерпевают кардинальные изменения. Одной из таких сфер является автомобильный транспорт и обслуживание автомобилей. С ростом числа автомобилей на дорогах, увеличивается и потребность в качественных автосервисах, которые могут удовлетворить потребности автовладельцев в ремонте, обслуживании и диагностике.

Актуальность темы данной дипломной работы заключается в том, что создание удобного и эффективного веб-приложения для поиска автосервисов может значительно упростить жизнь автовладельцам, предоставив им доступ к актуальной информации о ближайших автосервисах, их услугах, ценах и отзывах других клиентов. Кроме того, такое приложение может помочь автосервисам увеличить свою клиентскую базу, улучшить качество обслуживания и оптимизировать рабочие процессы.

Целью данной дипломной работы является разработка веб-приложения, которое будет обеспечивать эффективный поиск автосервисов, а также предоставление пользователям подробной информации о каждом автосервисе. Для достижения данной цели необходимо решить следующие задачи:

1. Проанализировать существующие аналоги веб-приложений для поиска автосервисов и выявить их недостатки.
2. Разработать архитектуру веб-приложения с учетом современных требований и стандартов веб-разработки.
3. Реализовать функционал поиска и фильтрации автосервисов по различным параметрам, таким как местоположение, предоставляемые услуги, стоимость и отзывы клиентов.
4. Обеспечить интеграцию с внешними сервисами для получения актуальной информации об автосервисах и отзывов клиентов.
5. Разработать удобный пользовательский интерфейс, который позволит пользователям быстро и легко находить необходимую информацию.
6. Протестировать разработанное веб-приложение на предмет его эффективности, надежности и удобства использования.

В ходе работы над проектом будут использоваться современные технологии, такие как HTML, CSS, JavaScript, React.js, Node.js, MongoDB, Bootstrap, jQuery и другие. Также будет уделено внимание вопросам безопасности и масштабируемости веб-приложения.

Разработанное веб-приложение для поиска автосервисов будет доступно для использования широкой аудитории автовладельцев, а также сможет стать полезным инструментом для автосервисов в продвижении своих услуг и улучшении качества обслуживания клиентов. Результаты данной дипломной работы могут быть использованы как в коммерческих, так и в некоммерческих целях, способствуя развитию отрасли автосервиса и улучшению качества жизни автовладельцев.

Описание проекта

Цель проекта

Целью проекта является разработка веб приложения на основе фронтенд фреймворка REACT и бэкенд фреймворка Django. Приложение должно удовлетворять требованиям и иметь минимальный необходимый функционал. Этот функционал должен включать:

- Регистрация пользователя в приложении
- Регистрация владельца автосервиса в приложении
- Возможность изменения данных пользователя в личном кабинете
- Возможность изменения данных владельца автосервиса в личном кабинете
- Возможность просмотра ближайших автосервисов пользователем
- Возможность отправки заявки на запись на обслуживание в автосервис
- Возможность просмотра заявок на ремонт владельцем автосервиса
- Возможность вести переписку с клиентом по отправленной заявке непосредственно в приложении
- Возможность изменять статус заявки
- Возможность закрывать заявку с различными статусами

Структура проекта

Проект размещен в нескольких репозиториях

- backend
- frontend
- gateway
- QA
- api-docs
- car-service
- tests

Основными репозиториями являются backend и frontend.

В репозитории frontend размещен код для фронтенд сайта. Разработкой фронтенд части приложения занимается команда фронтенд разработки.

Фронтенд приложения разработан с использованием технологий Type Script и REACT. Фронтенд часть развернута в Docker контейнере так как и Nginx сервер, который отвечает за передачу статического контента.

Как бэкенд разработчик, моя роль в проекте состоит в разработке API и бэкенд части приложения. Бэкенд часть приложения написана на языке программирования Python. Для создания приложения использован бэкенд фреймворк Django. Также в процессе работы на бэкенд частью использованы следующие технологии и инструменты:

Технологии и инструменты

- Linux - операционная система на которой работает приложение. Также я использую Linux как операционную систему на моей рабочей станции. Я использую Fedora Workstation.
- Docker - для размещения готового приложения на сервере и для облегчения разработки путем создания локальных контейнеров для тестирования работы как бэкенд так и фронтенд частей

приложения.
- GitHub Actions - для автоматизации развертывания приложения на сервере и организации CI-CD pipeline.
- Django REST framework - для создания API приложения.
- DRF spectacular - для создания документации к API приложения.
- Poetry - для управления зависимостями и сборки проекта.
- flake8 - линтер для Python кода
- black - для форматирования Python кода
- Pyright - Языковой сервер для авто-дополнения в редакторе кода и статического анализа Python кода.
- Neovim - мой редактор кода
- Postman - для тестирования API приложения.

В этой пояснительной записке далее я более подробно разберу использование каждого из этих инструментов и технологий.

Подготовка окружения для разработки приложения

Выбор операционной системы

Linux выбирают в качестве операционной системы для серверов в Интернете по следующим причинам:

- Открытость: Linux является открытым исходным кодом, что позволяет пользователям и разработчикам свободно изучать, модифицировать и распространять систему. Это обеспечивает гибкость и возможность адаптировать систему под индивидуальные потребности.
- Бесплатное распространение: Linux предлагается бесплатно, что снижает затраты на внедрение и поддержку сервера.
- Поддержка сообщества: Linux имеет огромное и активное сообщество разработчиков, пользователей и поставщиков, что обеспечивает поддержку и развитие системы.
- Безопасность: В Linux используется модель безопасности на основе привилегий, которая предотвращает несанкционированный доступ к системе. Кроме того, ядро Linux разработано таким образом, чтобы минимизировать уязвимости и эксплойты.
- Надежность: Linux зарекомендовал себя как стабильная и надежная операционная система для серверов. Он обеспечивает непрерывную работу и устойчивость к сбоям, что особенно важно для онлайн-сервисов и веб-приложений.
- Масштабируемость: Linux легко масштабируется для работы на различных типах серверов, от небольших и недорогих до высокопроизводительных и мощных. Это позволяет использовать Linux для различных задач, включая веб-хостинг, облачные вычисления, базы данных и многое другое.
- Совместимость: Linux поддерживает множество аппаратных и программных компонентов, что делает его совместимым с широким спектром оборудования. Это упрощает процесс развертывания и поддержки сервера.
- Гибкость и настраиваемость: Linux предлагает широкий выбор дистрибутивов и пакетов, что позволяет быстро и легко адаптировать систему для решения конкретных задач.

Варианты установки Linux для разработки приложения

Варианты установки Linux при разработке на этой платформе могут быть разными, в зависимости от требований к оборудованию, удобству использования и доступности инструментов. Вот несколько вариантов:

- Использование виртуальной машины (VM): Этот вариант подходит для тех, кто работает на платформе Windows или macOS и хочет протестировать свое приложение на Linux без необходимости установки этой операционной системы на свой компьютер. Виртуальная машина позволяет создать изолированную среду Linux, которую можно запускать на компьютере без влияния на основную операционную систему. Преимущества: простота использования, возможность легкого переключения между разными версиями Linux, совместимость с различными платформами. Недостатки: производительность может быть ниже, чем при использовании реальной установки Linux, некоторые приложения могут не работать должным образом.
- Установка Linux на отдельный жесткий диск или раздел: Этот вариант идеально подходит для тех, кому нужно постоянное и стабильное окружение для разработки на Linux. Преимущества: стабильная среда, возможность использовать все доступные инструменты Linux, высокая производительность. Недостатки: требуется больше времени на первоначальную настройку, нужен отдельный компьютер или отдельный раздел на компьютере.
- Использование облачной платформы: Некоторые облачные платформы, такие как Amazon Web Services (AWS), Microsoft Azure и Google Cloud Platform, предлагают предустановленные образы Linux для разработки и тестирования. Преимущества: возможность быстро развернуть среду Linux в облаке, масштабируемость, доступность инструментов для разработки. Недостатки: возможно, потребуется оплатить подписку на облако, производительность может зависеть от качества соединения с интернетом.
- Использование контейнеров: Это относительно новый подход к развертыванию и управлению приложениями, который позволяет упаковывать приложение и все его зависимости в один легкий и переносимый контейнер. Преимущества: быстрое развертывание и масштабирование, легкое обновление и миграция приложений. Недостатки: некоторые инструменты могут быть сложнее в использовании, чем виртуальные машины.

Установка нужной версии python

Проект основан на версии python 3.9.18. На Fedora 39 установлена версия python 12. Для того чтобы запустить версию python 3.9.18 необходимо использовать утилиту pyenv.

pyenv позволяет легко переключаться между несколькими версиями Python. Он прост, ненавязчив и следует традициям UNIX, когда инструменты одного назначения хорошо справляются с одной задачей.

Установка Poetry и настройка проекта

Poetry помогает объявлять, управлять и устанавливать зависимости для Python-проектов, гарантируя, что у вас везде будет правильный стек.

Поэзия заменяет setup.py, requirements.txt, setup.cfg, MANIFEST.in и Pipfile на простой формат проекта pyproject.toml.

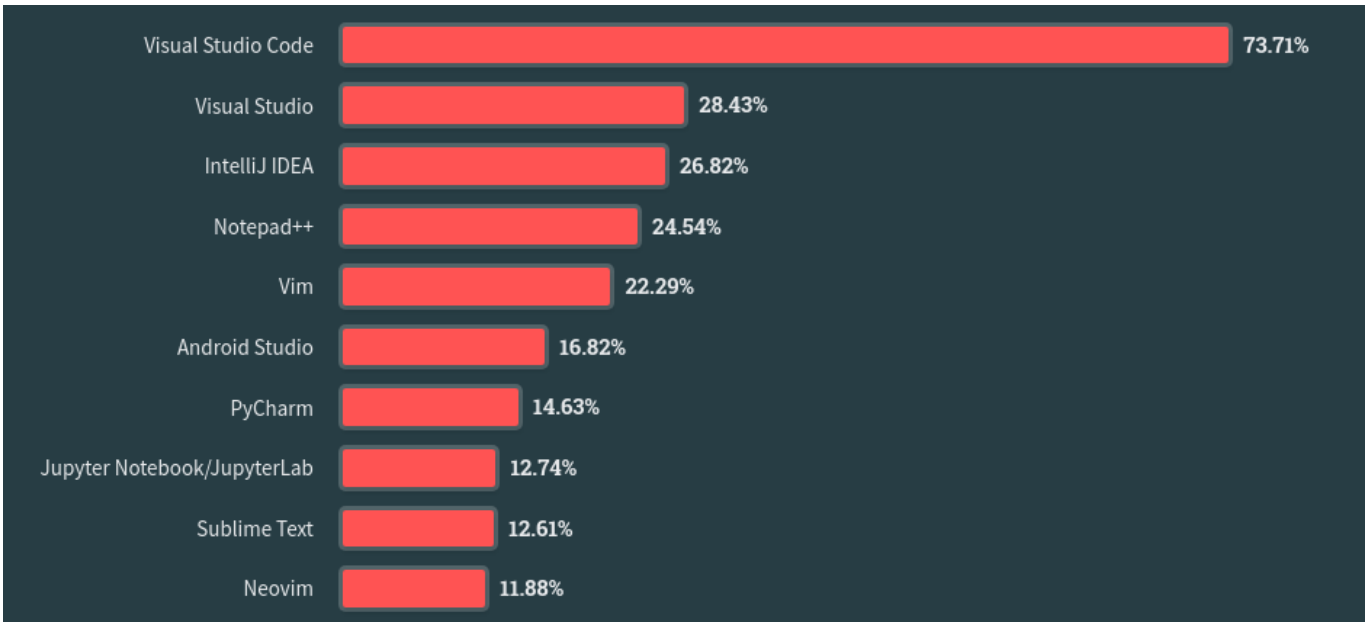
Установка дополнительных инструментов

Установка редактора

Каждый год на сайте StackOverflow проходит опрос пользователей сайта на темы связанные с разработкой программного обеспечения. В 2022 году в нем приняло участие более 74 тысячи человек. Stack Overflow survey

На вопрос какую среду разработки используют пользователи сайта были получены следующие результаты:

Visual Studio Code с большим отрывом опережает другие редакторы кода. Но интересно что редактор Vim занял пятое место и обошел другие более современные IDE.



Преимущества Vim:

Скорость редактирования кода

Визитной карточкой Vim является система горячих клавиш. В отличии от “обычных” редакторов текста, Vim использует систему режимов. В обычном режиме клавиши на клавиатуре используется для перемещения по тексту. Чтобы вводить текст нужно включить режим ввода текста.

Подобный подход позволяет отказаться от использования мыши для выделения и перемещения по тексту. Все действия выполняются с клавиатуры а руки не отрываются от среднего ряда клавиатуры.

В таком режиме выделение и редактирование текста происходит быстрее чем при использовании мыши. Например в случае если необходимо изменить текст внутри круглых скобок нужно набрать короткую фразу ci(. Текст внутри скобок будет удален и редактор перейдет в режим ввода текста.

Грамматика команд Vim может быть освоена в течении недели. После чего она поможет сэкономить гораздо большее времени затраченного на ее изучение. Использование горячих клавиш Vim в других программах

Клавиши Vim могут помочь не только при работе в самом редакторе Vim. Другие программы также имеют возможность использовать Vim режимы и клавиши. Zsh, Obsidian, Logseq, Emacs, Visual studio Code и продукты JetBrains - эти программы имеют возможность использовать горячие клавиши Vim в своей работе.

Если уметь работать в Vim даже при переходе на новую среду разработки можно будет продолжить использовать известные комбинации клавиш. Не придется тратить время чтобы запомнить либо переназначить клавиши в новой программе.

Близость к терминалу

Vim это консольное приложение. Это означает что оно запускается непосредственно в терминале. Что это значит для пользователя? Даже если ему придется работать с текстом на удаленном сервере, он сможет использовать Vim. Ему не придется устанавливать редактор, Vim или Vi предустановлены на большинстве Unix like систем.

Работа Vim в терминале означает что при работе с ним можно использовать любые команды из терминала.

- Можно вставить путь рабочей директории сразу в текст.
- Можно форматировать и фильтровать текст при помощи программы awk.
- Можно выполнять Git команды непосредственно в редакторе.

Vim позволяет по полной использовать coreutils.

Работа в связке с TMUX

В связке с мультиплексором терминалов VIm превращается в мощную среду разработки в которой можно работать над проектами, сохранять и восстанавливать состояние среды разработки между сессиями.

Тестирование и дебагинг кода может выполняться непосредственно в терминале. При этом переключение между редактором и процессом в консоли практически мгновенное.
Конфигурирование в текстовых файлах

Вим конфигурируется в текстовых файлах при помощи Vim script или Lua. По началу это может показаться странным тем пользователям, кто привык к выбору настроек в графическом интерфейсе. Но такой способ конфигурации является более простым и понятным. На пользователя не вываливается разом огромный набор возможных настроек. Пользователь сам выбирает какие настройки добавить в конфигурацию. Он может постепенно разобраться что дают ему выбранные им опции.

Изменения в текстовых файлах легко контролировать при помощи Git. В случае если изменение в настройках не устроило пользователя, оно может быть легко возвращено назад к предыдущему состоянию.

Большое число плагинов и расширений

В Vim есть большое число различных плагинов и расширений. Они расширяют функционал работы редактора и превращают его в полноценную среду разработки. LSP, completions, tree-sitter все это есть в Vim. В зависимости от потребностей пользователя Vim может быть как простым текстовым редактором так и не уступать в функциональности продуктам Jetbrains.

Установка языкового сервера и статического анализатора кода

В современном мире написание кода не представляется без использования статических анализаторов кода и языковых серверов.

Статический анализатор кода это программа которая парсит кодовую базу проекта и определяет ошибки типов, синтаксиса, логики в коде. Она показывает пользователю отчет с диагностическми данными и указывает как эти несоответсвия можно исправить.

Изначально статические анализаторы кода представляли собой cli инструмент который применяется из коммандной строки. Но с развитием инструментов, и со стремлением компании Microsoft продвигать свой редактор VS Code, этой компанией был разработан language server protocol. Этот протокол устанавливает стандартный способ взаимодействия между инструментами командной строки либо редакторами кода и языковым сервером.

Языкой сервер это программа которая обрабатывает код внутри файла или во всей кодовой базе и статически анализирует его. Клиентами же могут выступать редакторы кода инструменты командной строки и так далее.

Так language server protocol это opensorce технология, большинство современных редакторов кода поддерживают интеграцию с языковыми серверами.

Для python существует несколько языковых серверов. Одним из наиболее популярных является языковой сервер pyright разработанных компанией Microsoft.

Pyright можно выполнить несколькими способами. Установить локально в проект, установить глобально на систему, установить внутри редактора кода, где он будет доступен при редактировании.

В этом проекте я добавил его как development зависимость в файл pyproject.toml. Это позволит использовать его при необходимости из командной строки. Также это послужит своего рода документаций для других разработчиков о том, какие инструменты используются в проекте.

Текстовый редактор Neovim поддерживает протокол языкового сервера и может использовать языковой сервер pyright при работе для предоставления автодополнения, подсвечивания ошибок в коде, статического анализа и его визуализации внутри редактора, перемещения внутри файла и кодовой базы.

Для настройки статического анализатора кода используется файл pyproject.toml. Для этого нужно перечислить желаемые настройки в разделе `[tool.pyright]`.

Код настройки pyright

```
[tool.pyright]
include = []
exclude = [".pytest_cache",
    "**/__pycache__",
]
pythonVersion = "3.9.18"
typeCheckingMode = "standard"
```

Эти настройки озанчают:

- include - файлы или директории которые мы хотим явно включить в путь в котором работает pyright.
- exclude - файлы или директории которые мы хотим исключить пути. Это делается для улучшения скорости работы инструмента.
- pythonVersion - версия питон. В зависимости от версии python pyright будет указывать на устаревшие либо пока еще не используемые в данной версии питон синтаксические конструкции.
- typeCheckingMode - строгость к которой pyright будет подходить к анализу кода.

Установка линтера

Линтер это программа которая проверяет соответсвие написанного кода определенным правилам форматирования текста. Величина отступов, колличество пустых строк между параграфами и так далее.

Как и в случае с языковыми серверами существует несколько популярных линтеров для python. Некоторые из них это:

- flake8
- pylint
- ruff

Для данного проекта выбран линтер flake8.

flake8 проверяет код на соответствие стандарту pep8.

flake8 можно установить несколькими способами. Установить локально в проект, установить глобально на систему, установить внутри редактора кода, где он будет доступен при редактировании.

В этом проекте я добавил его как development зависимость в файл pyproject.toml. Это позволит использовать его при необходимости из командной строки. Также это послужит своего рода документаций для других разработчиков о том, какие инструменты используются в проекте.

Для настройки линтера кода используется файл pyproject.toml. Для этого нужно перечислить желаемые настройки в разделе `[tool.flake8]`.

Код настройки flake8

```
[tool.flake8]
max-line-length = 88
extend-ignore = ["E203", "I001", "I005", "R504"]
exclude = [
    ".git",
    "__pycache__",
    "env",
    "migrations",
    "settings.py",
    "venv",
    "management"
]
max-complexity = 10
```

max-line-length - Устанавливает максимальную длину, которую может иметь любая строка (за некоторыми исключениями). extend-ignore - Указывает список кодов для добавления в список игнорируемых. exclude - Укажите список глобальных паттернов, разделенных запятыми, чтобы исключить их из проверок. max-complexity - Устанавливает максимально допустимое значение сложности McCabe для блока кода.

Установка форматера

Если линтер только проверяет

Установка проекта

Файл pyproject.toml