FLASK КАК ИНСТРУМЕНТ РАЗРАБОТКИ ИНТЕЛЛЕКТУАЛЬНЫХ ВЕБ-СИСТЕМ

Ёвженко Юрий Дмитриевич
студент,
Гомельский Государственный Технический университет имени П.О. Сухого,
Республика Беларусь, г. Гомель

Житко Алина Сергеевна
студент,
Гомельский Государственный Технический университет имени П.О. Сухого,
Республика Беларусь, г. Гомель

В современном мире цифровых технологий веб-фреймворки играют ключевую роль в разработке разнообразных приложений. Среди них особое место занимает Flask – Python-микрофреймворк, известный своей простотой, гибкостью и мощной экосистемой. Цель данной работы – продемонстрировать, почему Flask является превосходным выбором для создания проектов в области машинного зрения, а также для разработки сложных экспертных веб-систем, на примере системы по созданию планограмм. Актуальность применения этих технологий в бизнесе постоянно растет, открывая новые возможности для автоматизации и повышения эффективности.

Ключевые преимущества фреймворка. Одним из ключевых достоинств Flask является его минимализм. В отличие от более монолитных фреймворков, Flask предоставляет лишь ядро, необходимое для создания веб-приложения, оставляя разработчику полную свободу в выборе инструментов и компонентов, будь то система управления базами данных (ORM), шаблонизатор или механизмы аутентификации. Эта гибкость особенно ценна, когда речь идет об интеграции существующего Python-кода, что часто встречается в проектах машинного зрения или при разработке сложных алгоритмов для экспертных систем.

Простота изучения и использования Flask делает его привлекательным как для новичков, так и для опытных разработчиков [1]. Низкий порог входа и лаконичный синтаксис позволяют быстро приступить к разработке и создавать

прототипы в сжатые сроки. Однако настоящая мощь Flask раскрывается благодаря экосистеме Python, т.к. разработчики получают доступ ко всему богатству Python-библиотек, что является критически важным аргументом.

Несмотря на свою легковесность, Flask обладает высокой расширяемостью. Существует огромное количество готовых расширений для решения типичных задач, таких как работа с базами данных (Flask-SQLAlchemy), создание RESTful API (Flask-RESTful) или управление аутентификацией пользователей (Flask-Login). При необходимости разработчик может легко создать и собственные расширения, адаптируя фреймворк под специфические нужды проекта.

С точки зрения производительности, Flask демонстрирует достойные результаты для большинства веб-приложений, особенно когда речь идет о создании API-сервисов. Для высоконагруженных или ресурсоемких задач всегда есть возможность интеграции с асинхронными решениями, например, через ASGI-серверы или используя системы очередей задач вроде Celery для выполнения фоновых операций. Легкость создания RESTful API делает Flask отличным кандидатом для бэкенда, обслуживающего как веб-интерфейсы, так и мобильные приложения или другие сервисы.

Flask для проектов машинного зрения (CV). Обоснование выбора Flask для проектов в области машинного зрения кроется, прежде всего, в его тесной связи с Python. Это позволяет осуществлять прямую интеграцию с Python-библиотеками CV без необходимости сложных оберток или межпроцессного взаимодействия. Разработчики могут быстро создавать веб-интерфейсы для демонстрации работы своих CV-моделей: например, страница, куда пользователь загружает изображение или видео, а система отображает результаты анализа.

Кроме того, Flask идеально подходит для разработки API, предоставляющих доступ к CV-моделям как к сервису (MlaaS-подход). Это позволяет другим системам или приложениям легко использовать функциональность машинного зрения, отправляя данные на обработку и получая результаты [2, с. 185]. Flask эффективно справляется с обработкой HTTP-запросов, содержащих изображения или даже видеопотоки. Типичные сценарии использования включают

сервисы распознавания объектов на изображениях, системы анализа видеопотока с камер наблюдения в реальном времени или веб-приложения для интерактивной разметки данных, необходимых для обучения CV-моделей.

Архитектурно, при работе с машинным зрением на Flask, важно учитывать передачу и обработку потенциально больших объемов данных. Например, система может быть спроектирована так, что Flask-приложение принимает HTTP-запрос с изображением, передает его в отдельный модуль, использующий OpenCV или TensorFlow для анализа, а затем возвращает результат клиенту [3]. Для ресурсоемких задач обработки, которые могут занимать продолжительное время, целесообразно использовать асинхронную обработку с помощью Celery, чтобы не блокировать основной поток веб-сервера и обеспечить отзывчивость интерфейса.

Flask для создания экспертной веб-системы по созданию планограмм. Планограмма – это визуальное представление оптимального размещения товаров на полках в торговой точке. Создание эффективных планограмм требует учета множества факторов: данных о продажах, характеристик товаров, маркетинговых стратегий, правил мерчандайзинга и совместимости продуктов. Экспертная веб-система призвана автоматизировать и оптимизировать этот процесс, и Flask прекрасно подходит для создания таких систем.

Ключевым преимуществом здесь является гибкость Python в реализации сложной бизнес-логики, которая позволяет эффективно кодировать правила и алгоритмы, составляющие ядро экспертной системы. Это становится особенно актуальным, когда экспертная система должна интерпретировать и применять правила, заданные в текстовом формате, с использованием больших языковых моделей (LLM) или собственной системы парсинга правил, для чего может быть разработан специальный метаязык экспертной системы. Вместо того чтобы полагаться исключительно на жестко закодированные ограничения, система может анализировать текстовые инструкции мерчандайзеров или маркетинговые директивы. Python, с его богатой экосистемой библиотек для работы с LLM (таких как transformers, библиотеки для взаимодействия с OpenAI API или Langchain), позволяет Flask-приложению легко интегрировать такой интеллектуальный

анализ. Flask может служить интерфейсом для ввода этих текстовых правил, а затем передавать их на обработку LLM или специализированному парсеру в бэкенде [2, с. 185, 415]. Они, в свою очередь, преобразуют неструктурированный текст в структурированные данные, логические условия или весовые коэффициенты, которые затем используются системой для генерации планограмм. При этом традиционные, более формализованные правила, такие как запрет размещения определенных категорий товаров рядом или предписание совместной выкладки комплементарных продуктов для стимулирования продаж, также легко реализуются и могут работать в синергии с выводами такой системы.

Интеграция с базами данных – еще один сильный аспект. Flask легко подключается к различным СУБД для хранения обширной информации о товарах, их атрибутах, истории планограмм. Это формирует надежную основу для принятия системой обоснованных решений, которые могут быть комбинацией как детерминированных правил, так и выводов, сделанных искусственным интеллектом.

Для создания пользовательского интерфейса Flask предлагает несколько подходов. Можно использовать встроенный шаблонизатор Jinja2 для серверного рендеринга интерфейса управления планограммами, где пользователи смогут просматривать и редактировать схемы, а также вводить или загружать текстовые правила для последующего анализа системой. В качестве альтернативы, Flask может выступать как backend API для современного frontend-фреймворка, который будет отвечать за создание визуального редактора планограмм, где пользователь может визуально манипулировать товарами на полках или ввести текстовые директивы.

Архитектурно такая система на Flask может состоять из нескольких ключевых компонентов: модуля управления каталогом товаров, модуля управления и анализа правил размещения, визуального редактора планограмм, а также API для возможной интеграции с другими корпоративными системами, например, ERP или системами управления запасами. Flask в такой архитектуре выступает как связующее звено, обеспечивая взаимодействие между компонентами и предоставляя интерфейсы для пользователей и других систем.

Заключение. Flask, благодаря своей минималистичной природе, гибкости и тесной интеграции с богатой экосистемой Python, представляет собой исключительно мощный и удобный инструмент для широкого круга задач. Как было показано, он превосходно подходит для разработки сервисов машинного зрения, позволяя легко интегрировать сложные CV-модели и предоставлять к ним доступ через веб-интерфейсы или API. Одновременно с этим, его возможности позволяют создавать сложные экспертные веб-системы, такие как системы для автоматизированного формирования планограмм, где требуется реализация нетривиальной бизнес-логики и работа с большими объемами данных. Таким образом, Flask уверенно занимает свою нишу, предоставляя разработчикам надежную основу для создания интеллектуальных и эффективных веб-решений.

Список литературы:

1. Flask: Building Python Web Services / G. Dwyer [et al.]. – Packt Publishing, 2017. – 770 p.
2. Просиз, Д. Прикладное машинное обучение и искусственный интеллект для инженеров / Д. Просиз. – Астана: АЛИСТ, 2024. – 432 с.
3. Yaganteeswarudu, A. Multi Disease Prediction Model by using Machine Learning and Flask API / A. Yaganteeswarudu // 5th International Conference on Communication and Electronics Systems (ICCES). – Coimbatore, 2020. – P. 1242-1246.