Открытый журнал евразийских исследований, 2026, №2, сс. 116-127
doi: 10.65469/eijournal.2026.2.14
eijournal.ilimbilim.kg

ИНФОРМАТИКА / COMPUTER SCIENCE
УДК 004.04

Модульная архитектура веб-приложений для автоматизации управления кофе-ресторанов на основе Django Framework

Аркабаев Нуркасым Кылычбекович

к. ф.-м.н., доцент, Ошский государственный университет, Кыргызстан, narkabaev@oshsu.kg, ORCID: 0009-0000-1912-2225

Аскарова Лилия Сиюрбековна

магистрант, Ошский государственный университет, Кыргызстан, askarovaliliyao8@gmail.com, ORCID: 0009-0005-2060-4519

Айтматова Гулзада Досматовна

магистрант, Ошский государственный университет, Кыргызстан, aitmatovao101@gmail.com, ORCID: 0009-0008-6783-6431

Аннотация

В статье рассматривается проектирование и реализация модульной архитектуры веб-приложения для автоматизации управления кофе-ресторанами на основе Django Framework. Проведён анализ существующих систем автоматизации ресторанного бизнеса (R-Keeper, iiko, Poster POS), выявлены их ограничения для малого и среднего бизнеса в условиях Кыргызстана. Предложена шестимодульная архитектура системы «Ресторан», включающая модули управления филиалами, персоналом, меню, складом, заказами и аналитикой. Описаны принципы декомпозиции системы на независимые Django-приложения, схема взаимодействия модулей через RESTful API, модели данных и механизм автоматического списания ингредиентов. Результаты тестирования подтверждают работоспособность системы и её соответствие функциональным требованиям предметной области.

Ключевые слова: модульная архитектура, веб-приложение, Django Framework, автоматизация ресторанного бизнеса, кофе-ресторан, RESTful API, управление филиалами

Для цитирования: Аркабаев Н.К., Аскарова Л.С., Айтматова Г.Д. (2026). Модульная архитектура веб-приложений для автоматизации управления кофе-ресторанов на основе Django Framework. *Открытый журнал евразийских исследований*, №2, сс. 116-127. doi: 10.65469/eijournal.2026.2.14

Введение

В наши дни сеть Интернет глубоко интегрировалась в повседневную жизнь людей и стала важнейшей средой для поддержки бизнес-процессов. С развитием технологий возрастает потребность в веб-приложениях, обеспечивающих высокую скорость отклика на

действия пользователя и корректную работу на различных устройствах как настольных, так и мобильных [1, 2, 10, 11].

Индустрия общественного питания в Центральной Азии переживает период интенсивного роста, сопровождающегося усложнением операционных процессов ресторанного бизнеса. По данным аналитических обзоров, оборот заведений общественного питания в регионе ежегодно увеличивается на 12-15%, при этом сетевые форматы демонстрируют опережающую динамику [3]. Управление сетью кофе-ресторанов требует координации множества бизнес-процессов: от учёта рабочего времени сотрудников до контроля расходования ингредиентов на каждом филиале. Попытки решить эти задачи вручную или с помощью разрозненных инструментов приводят к потере оперативности, ошибкам в учёте и снижению качества обслуживания. Во-вторых, разработка веб-приложений является одним из наиболее востребованных направлений в IT-индустрии [4].

На рынке представлены зарекомендовавшие себя системы автоматизации – R-Keeper, iiko, Poster POS и ряд международных платформ (Toast, Square for Restaurants). Тем не менее при попытке внедрения этих решений в условиях КР обнаруживается ряд ограничений: высокая стоимость лицензий и серверного оборудования, отсутствие адаптации к местному законодательству, языковые барьеры, а также недостаточная гибкость при работе с небольшими сетями из 3-7 филиалов [5].

Перечисленные проблемы формируют запрос на разработку доступного, адаптированного к региональной специфике веб-приложения, которое можно было бы внедрять поэтапно и расширять по мере роста бизнеса. Архитектурный подход к проектированию такого приложения приобретает ключевое значение: от того, насколько грамотно система разделена на модули, зависит и скорость разработки, и возможности масштабирования, и затраты на сопровождение.

Цель данной статьи – описать проектирование и реализацию модульной архитектуры веб-приложения «Ресторан» для автоматизации управления сетью кофе-ресторанов на основе Django Framework и оценить работоспособность предложенного архитектурного решения.

Для достижения поставленной цели решались следующие задачи: проведение анализа существующих решений с выявлением их функциональных и экономических ограничений; обоснование выбора технологического стека; проектирование шестимодульной архитектуры с определением зависимостей между модулями; реализация ключевых компонентов системы; тестирование и оценка результатов.

Материал и методы исследования

Выбор фреймворка для серверной части определялся совокупностью функциональных, нефункциональных и экономических критериев, специфичных для предметной области ресторанного бизнеса. Сравнительному анализу подвергались четыре фреймворка: Django, Flask, FastAPI и Spring Boot. Каждый из них оценивался по девяти критериям с весовыми коэффициентами, отражающими приоритеты проекта (таблица 1).

Таблица 1. Критерии оценки веб-фреймворков

Группа критериев	Критерий	Вес	Описание
Функциональные	Скорость разработки MVP	0,20	Быстрое создание прототипа
	Встроенная ORM	0,15	Работа с базами данных
	Система аутентификации	0,10	Готовые решения безопасности

	Административный интерфейс	0,15	Создание админ-панели
Нефункциональные	Производительность	0,10	Скорость обработки запросов
	Масштабируемость	0,10	Поддержка роста нагрузки
	Безопасность	0,10	Защита от веб-атак
Экономические	Стоимость разработки	0,05	Время и ресурсы
	Доступность разработчиков	0,05	Наличие специалистов

Django получил наивысшую взвешенную оценку (8,5 из 10) за счёт встроенной административной панели, полноценной ORM, системы аутентификации и авторизации, а также развитой экосистемы пакетов. Flask и FastAPI уступили в части готовой инфраструктуры для управления данными (отсутствие административного интерфейса, необходимость самостоятельной настройки ORM). Spring Boot, обладая высокой производительностью, требует значительно большего времени на начальное развёртывание и конфигурирование, что критично при ограниченных ресурсах разработки.

Таблица 2. Сравнительная характеристика веб-фреймворков

Критерий	**Django**	**Flask**	**FastAPI**	**Spring Boot**
Скорость разработки	Высокая	Средняя	Высокая	Низкая
Админ-интерфейс	Встроенный	Нет	Нет	Настраиваемый
ORM из коробки	Да	Нет	Нет	Да
Безопасность	Высокая	Настраиваемая	Средняя	Высокая
Подходит для MVP	Да	Частично	Для API	Нет
Итоговая оценка	8,5/10	6,0/10	7,0/10	6,5/10

Полный технологический стек системы «Ресторан» включает: Python 3.11+ и Django 5.2 на серверной стороне, Django REST Framework 3.14 для построения API, PostgreSQL 15 в качестве основной СУБД (SQLite – для локальной разработки), HTML5/CSS3/JavaScript и Bootstrap 5.3 на клиентской стороне, Redis для кэширования сессий, а также Git для контроля версий. Выбор PostgreSQL обусловлен поддержкой сложных аналитических запросов, полнотекстового поиска и транзакционной целостности, критичных для складского и финансового учёта.

Django реализует паттерн MVT (Model-View-Template), адаптирующий классическую схему MVC к специфике веб-разработки. Компонент Model инкапсулирует бизнес-логику и структуру данных, View обрабатывает HTTP-запросы и формирует ответы, а Template отвечает за рендеринг HTML-страниц. Разделение ответственности между этими компонентами обеспечивает тестируемость, поддерживаемость кода и возможность параллельной работы над различными уровнями приложения [6].

Ключевым архитектурным решением при проектировании системы «Ресторан» стало разбиение функциональности на шесть независимых Django-приложений, каждое из которых представляет собой логически обособленный домен предметной области. Принципы декомпозиции основывались на концепции ограниченных контекстов (bounded contexts) из предметно-ориентированного проектирования (DDD): каждый модуль владеет собственными моделями данных, представлениями, шаблонами и URL-маршрутами, а взаимодействие между модулями происходит через чётко определённые интерфейсы – внутренние импорты моделей и RESTful API [7, 8, 9].

Модули системы «Ресторан» организованы следующим образом: accounts (управление пользователями и ролевой моделью доступа), restaurants (администрирование филиалов сети), menu (управление категориями блюд, позициями меню, рецептами и ценообразованием), staff (кадровый учёт и привязка сотрудников к филиалам), inventory (складской учёт – ингредиенты, остатки, рецептурные связи), orders (обработка заказов с автоматическим списанием ингредиентов), analytics (генерация отчётов и аналитических данных).

Результаты и обсуждение

Анализ рынка систем автоматизации ресторанного бизнеса охватил три наиболее распространённых в постсоветском пространстве решения: R-Keeper, iiko и Poster POS. Каждая система обладает определёнными преимуществами, однако ни одна из них полностью не удовлетворяет требованиям небольших ресторанных сетей в КР.

R-Keeper, функционирующий с 1992 года, обеспечивает высокую надёжность и масштабируемость для крупных сетей, однако характеризуется устаревшим интерфейсом, требующим привлечения программиста для настройки, и высокой стоимостью внедрения (200-300 тысяч сом). Базовая версия ограничена функционалом фронт-офиса, а для складского учёта необходим отдельный модуль StoreHouse. Система iiko, появившаяся в 2005 году, предлагает более современный интерфейс и комплексное решение «из коробки», но при переходе к сетевой версии требует замены данных и повторной настройки, что влечёт дополнительные затраты. Poster POS ориентирован на малый бизнес и отличается низкой стоимостью (40-50 тысяч сом), но ограничен в части аналитики, масштабируемости и возможности управления несколькими филиалами из единой точки

Таблица 3. Сравнительный анализ существующих систем автоматизации

Параметр	R-Keeper	iiko	Poster POS	«Ресторан»
Стоимость внедрения	Высокая	Средняя	Низкая	Низкая
Сложность настройки	Высокая	Средняя	Низкая	Низкая
Масштабируемость	Отличная	Хорошая	Ограничена	Хорошая
Интерфейс	Устаревший	Современный	Современный	Современный
Адаптация к КР	Нет	Частично	Нет	Да
Модульное расширение	Ограничено	Ограничено	Нет	Да
Открытый стек	Нет	Нет	Нет	Да

Как видно из таблицы 3, разрабатываемая система «Ресторан» обладает рядом преимуществ для целевого сегмента – небольших ресторанных сетей в Кыргызстане: низкая стоимость владения благодаря использованию open-source-технологий, адаптация к местной специфике и возможность поэтапного расширения функциональности через подключение новых модулей.

Архитектура системы «Ресторан» построена на принципе минимальных зависимостей между модулями. Модуль accounts не зависит ни от какого другого модуля и предоставляет сервисы аутентификации и авторизации всем остальным компонентам. Модуль restaurants зависит только от accounts и определяет сущность филиала, к которой привязываются сотрудники, складские позиции и заказы. Модули staff, menu и inventory зависят от restaurants, но не зависят друг от друга напрямую. Модуль orders занимает верхний уровень иерархии

зависимостей: он связывает menu (позиции заказа), restaurants (принадлежность заказа филиалу), accounts (автор заказа) и inventory (списание ингредиентов).

Таблица 4. Зависимости и API-эндпоинты модулей системы «Ресторан»

Модуль	Зависимости	Предоставляемые сервисы	API endpoints
accounts	-	Аутентификация, авторизация, роли	/api/auth/, /api/users/
restaurants	accounts	Управление филиалами	/api/branches/
staff	accounts, restaurants	Кадровый учёт	/api/staff/
menu	restaurants	Категории, блюда, рецепты	/api/menu/, /api/categories/
inventory	restaurants, menu	Складской учёт, остатки	/api/inventory/, /api/stock/
orders	accounts, restaurants, menu, inventory	Заказы, списание ингредиентов	/api/orders/
analytics	все модули	Отчёты, дашборд, KPI	/api/analytics/, /api/reports/

Такая иерархия зависимостей обеспечивает несколько практических преимуществ. Во-первых, каждый модуль может разрабатываться и тестироваться автономно – достаточно подключить его зависимости через фикстуры или моки. Во-вторых, замена или рефакторинг одного модуля (например, переход от простого складского учёта к интеграции с внешней ERP-системой) не затрагивает остальные компоненты при условии сохранения интерфейсов. В-третьих, при масштабировании отдельные модули могут быть вынесены в микросервисы, поскольку взаимодействие уже происходит через API.

Рассмотрим основные модели данных, реализованные в системе. Модель пользователя (CustomUser) расширяет стандартный класс AbstractUser Django, добавляя поле role с тремя значениями: ADMIN (администратор системы), MANAGER (менеджер филиала) и STAFF (рядовой сотрудник). Аутентификация осуществляется по адресу электронной почты (поле USERNAME_FIELD = 'email'), что упрощает управление учётными записями в распределённой сети филиалов.

Модель MenuItem включает помимо стандартных атрибутов (название, описание, цена, категория) расширенные поля: время приготовления, калорийность, вес, признаки «острое» и «вегетарианское», себестоимость. Наличие поля cost_price позволяет автоматически рассчитывать маржу прибыли через свойство profit_margin. Связь блюда с ингредиентами реализована через модель Recipe, определяющую количество каждого ингредиента на одну порцию.

Модель Order реализует жизненный цикл заказа через четыре статуса: PENDING → IN_PROGRESS → COMPLETED / CANCELLED. При переходе заказа в статус IN_PROGRESS срабатывает сигнал Django (post_save), инициирующий автоматическое списание ингредиентов со склада соответствующего филиала. Метод process_ingredients() использует атомарную транзакцию (transaction.atomic), обеспечивая целостность данных: если списание хотя бы одного ингредиента невозможно из-за отсутствия на складе, система фиксирует предупреждение, списывает доступный остаток и продолжает обработку. Такой подход предотвращает блокировку обслуживания при недостаточном количестве отдельных ингредиентов.

Одним из технически наиболее сложных компонентов системы является механизм автоматического списания ингредиентов при обработке заказов. Алгоритм работает следующим образом. При изменении статуса заказа на IN_PROGRESS сигнал post_save

вызывает метод process_ingredients(). Метод итерирует по всем позициям заказа (OrderItem), для каждой позиции извлекает рецептуру (Recipe), рассчитывает необходимое количество ингредиента с учётом количества порций и выполняет операцию списания из таблицы StockItem, привязанной к конкретному ресторану. Вся операция заключена в блок transaction.atomic(), что гарантирует откат при возникновении ошибок. Если на складе недостаточно ингредиента, система списывает весь доступный остаток и добавляет предупреждение в массив warnings, не прерывая обслуживание.

Флаг ingredients_processed в модели Order предотвращает повторное списание при множественных сохранениях объекта заказа. Такое архитектурное решение обеспечивает идемпотентность операции – многократный вызов process_ingredients() для одного заказа не приводит к избыточному списанию.

Система реализует трёхуровневую ролевую модель доступа: администратор системы имеет полный доступ ко всем филиалам и модулям; менеджер филиала – к данным закреплённого за ним филиала с возможностью управления персоналом, меню и складом; рядовой сотрудник – к интерфейсу обработки заказов и просмотру личного графика. Проверка доступа реализована через методы has_admin_access() и has_manager_access() модели CustomUser, которые используются в декораторах представлений и permission-классах Django REST Framework.

Клиентская часть построена на базе Bootstrap 5.3 с использованием адаптивной сетки и компонентов. Центральным элементом интерфейса является дашборд, отображающий ключевые метрики: количество активных филиалов, общее число сотрудников, объём заказов за текущий период и уведомления о критических уровнях складских запасов. Класс DashboardManager на JavaScript обеспечивает периодическое обновление данных через асинхронные fetch-запросы с интервалом 30 секунд, анимированное обновление числовых показателей и обработку пользовательских событий.

Адаптивность интерфейса обеспечивает корректное отображение на устройствах с различными разрешениями экрана – от настольных мониторов до смартфонов. На мобильных устройствах реализована упрощённая навигация через collapse-меню, увеличенные области касания для интерактивных элементов и горизонтальная прокрутка таблиц. Формы ввода данных используют прогрессивную валидацию на трёх уровнях: клиентская JavaScript-валидация, серверная Django-валидация и бизнес-логическая валидация правил предметной области.

При проектировании архитектуры системы были применены несколько классических паттернов проектирования, адаптированных к специфике Django Framework. Паттерн Repository реализован через менеджеры моделей Django (Model Managers), инкапсулирующие типовые запросы к базе данных. Например, менеджер модели StockItem включает методы для получения позиций с критическим уровнем остатков и для формирования отчётов по движению товаров за указанный период. Такой подход позволяет не дублировать логику запросов в представлениях и сериализаторах.

Паттерн Observer реализован через встроенный механизм сигналов Django. В системе «Ресторан» сигнал post_save модели Order используется для автоматического запуска процедуры списания ингредиентов при изменении статуса заказа. Аналогичным образом сигналы применяются для генерации уведомлений о критических уровнях складских запасов – при каждом обновлении модели StockItem проверяется, не опустился ли остаток ниже порогового значения, и при необходимости формируется запись в таблице уведомлений.

Паттерн Strategy применяется при расчёте себестоимости блюд: различные алгоритмы калькуляции (по фактическим ценам закупки, по средневзвешенным ценам, по последней закупочной цене) реализованы как отдельные функции, выбираемые через конфигурацию филиала. Это обеспечивает гибкость при адаптации системы к различным учётным политикам заведений.

Для обеспечения взаимодействия между модулями и предоставления интерфейса для внешних интеграций в системе «Ресторан» используется Django REST Framework (DRF). API построен с соблюдением принципов REST: каждый ресурс (ресторан, блюдо, заказ, сотрудник) адресуется через уникальный URL, стандартные HTTP-методы (GET, POST, PUT, DELETE) определяют характер операции, а статусные коды ответов соответствуют семантике операций.

Сериализаторы DRF обеспечивают валидацию входящих данных и преобразование моделей Django в JSON-представление. Для модели MenuItem, например, сериализатор автоматически рассчитывает и включает в ответ вычисляемые поля profit_margin и profit_amount, не хранящиеся в базе данных. Permission-классы обеспечивают контроль доступа на уровне API: эндпоинты аналитики доступны только пользователям с ролью ADMIN, операции управления персоналом – пользователям с ролями ADMIN и MANAGER, а просмотр меню и создание заказов – всем авторизованным пользователям.

Версионирование API реализовано через URL-префиксы (/api/v1/), что позволяет в будущем вводить новые версии без нарушения работы существующих клиентов. Документация API генерируется автоматически на основе конфигурации DRF и доступна через встроенный browsable API интерфейс, упрощающий отладку и тестирование.

Стратегия тестирования системы «Ресторан» включает три уровня: модульное тестирование моделей и бизнес-логики, интеграционное тестирование API-эндпоинтов и функциональное тестирование пользовательских сценариев. Модульные тесты проверяют корректность работы отдельных компонентов: создание и связывание сущностей (филиалы, сотрудники, блюда), расчёт себестоимости и маржи, логику списания ингредиентов и корректность ролевой модели доступа.

Интеграционные тесты используют встроенный в Django тестовый клиент для проверки API-эндпоинтов: правильность HTTP-статусов ответов, корректность сериализации и десериализации данных, соблюдение ограничений доступа для различных ролей. Функциональные тесты через Selenium WebDriver автоматизируют проверку основных пользовательских сценариев: авторизация, создание заказа, проверка корректности дашборда и работы адаптивной навигации на различных разрешениях экрана.

Тестирование механизма автоматического списания ингредиентов заслуживает отдельного внимания, поскольку эта функциональность связана с финансовым учётом. Тестовые сценарии включают проверку корректности списания при наличии всех ингредиентов на складе, обработку ситуации недостаточного количества ингредиента, проверку идемпотентности (повторный вызов не приводит к двойному списанию), а также тестирование атомарности транзакции при возникновении ошибок.

Таблица 5. Результаты тестирования системы «Ресторан»

Модуль	Тестов	Пройдено	Ошибок	Покрытие, %
accounts (аутентификация)	18	18	0	92
restaurants (филиалы)	12	12	0	88

menu (меню и рецепты)	24	24	0	85
staff (персонал)	10	10	0	80
inventory (склад)	20	20	0	90
orders (заказы, списание)	28	28	0	94
analytics (аналитика)	8	8	0	75
Итого	120	120	0	86

Все 120 тестов были пройдены успешно. Среднее покрытие кода тестами составило 86%, при этом наиболее критичные модули (orders и accounts) имеют покрытие 94% и 92% соответственно. Тестирование производительности показало, что среднее время ответа API-эндпоинтов не превышает 200 мс при одновременной работе до 50 пользователей, что соответствует нефункциональным требованиям системы.

Предварительные расчёты экономической эффективности внедрения системы «Ресторан» основаны на экспертных оценках и данных о типичных затратах ресторанной сети из 5 филиалов. Стоимость внедрения разработанного решения (включая развёртывание, настройку и обучение персонала) оценивается в 80-120 тысяч сомов, что в 3-5 раз ниже стоимости внедрения R-Keeper или iiko для аналогичной конфигурации. Ежемесячные затраты на обслуживание (хостинг, резервное копирование, техническая поддержка) составляют порядка 5-8 тысяч сомов.

Ожидаемый эффект от внедрения включает сокращение времени на выполнение административных операций на 40-50% за счёт автоматизации складского учёта, управления персоналом и аналитической отчётности; повышение точности складского учёта до 95% благодаря автоматическому списанию ингредиентов при обработке заказов; снижение потерь от порчи продуктов на 15-20% за счёт своевременного контроля остатков и уведомлений о критических уровнях запасов. При указанных параметрах расчётный срок окупаемости системы составляет 6-10 месяцев.

Выводы

Результаты проектирования и реализации системы «Ресторан» позволяют сформулировать несколько выводов о применимости модульной архитектуры на основе Django Framework для автоматизации управления кофе-ресторанами.

Во-первых, декомпозиция системы на шесть независимых Django-приложений (accounts, restaurants, menu, staff, inventory, orders) с чётко определёнными зависимостями обеспечила возможность параллельной разработки и автономного тестирования каждого модуля. Модульный подход позволил поэтапно наращивать функциональность: базовая версия системы включала только модули accounts, restaurants и menu, а модули inventory, orders и analytics добавлялись на последующих итерациях без необходимости рефакторинга существующего кода.

Во-вторых, механизм автоматического списания ингредиентов, реализованный через сигналы Django и атомарные транзакции PostgreSQL, продемонстрировал устойчивость при обработке конкурентных заказов. Идемпотентность операции, обеспечиваемая флагом ingredients_processed, исключает ошибки двойного списания даже при повторных сохранениях объекта заказа.

В-третьих, использование Django REST Framework для построения API между модулями создало основу для дальнейшей эволюции архитектуры – потенциального выделения

отдельных модулей в микросервисы при росте нагрузки, а также для интеграции с внешними системами (бухгалтерскими программами, сервисами доставки, мобильными приложениями).

В-четвёртых, трёхуровневая ролевая модель доступа (администратор-менеджер-сотрудник) адекватно отражает организационную структуру ресторанной сети и обеспечивает разграничение полномочий без избыточной сложности.

Представленная в статье модульная архитектура веб-приложения «Ресторан» на основе Django Framework решает задачу автоматизации управления сетью кофе-ресторанов с учётом специфики малого и среднего бизнеса Кыргызстана. Шестимодульная структура системы, построенная на принципах минимальных зависимостей и чётких интерфейсов между компонентами, обеспечивает как текущую функциональность (управление филиалами, персоналом, меню, складом, заказами и аналитикой), так и возможности расширения по мере роста бизнеса.

Выбор Django в качестве основного фреймворка оправдан для данного класса задач: встроенная ORM, административная панель, система аутентификации и Django REST Framework существенно сократили время разработки и позволили сосредоточиться на бизнес-логике предметной области. Вместе с тем следует отметить, что синхронная природа Django может стать ограничивающим фактором при значительном росте числа одновременных подключений (свыше 500). В этом случае целесообразно рассмотреть частичный переход на асинхронную обработку запросов через Django Channels или выделение высоконагруженных эндпоинтов в отдельные сервисы на FastAPI.

Среди ограничений текущей реализации следует выделить отсутствие интеграции с внешними кассовыми аппаратами и фискальными системами, что необходимо для полного соответствия налоговому законодательству Кыргызской Республики. Кроме того, система не включает модуль управления доставкой, востребованный в современном ресторанном бизнесе.

Направления дальнейшего развития включают: интеграцию с фискальными системами КР; разработку мобильного приложения на базе существующего REST API; внедрение модуля прогнозирования спроса на основе анализа исторических данных о заказах; добавление модуля управления лояльностью клиентов; оптимизацию производительности через внедрение кэширования Redis и асинхронной обработки фоновых задач через Celery.

Практическое внедрение системы в сети кофе-ресторанов «Ресторан» позволит верифицировать архитектурные решения в реальных условиях эксплуатации и собрать данные для количественной оценки экономического эффекта автоматизации.

Список литературы

1. Потовиченко, М. А., Шатилов, Ю. Ю. (2020). Разработка клиентской части одностраничного Web-приложения с использованием библиотеки React. Научное обозрение. Технические науки, 1, 39-43. URL: https://science-engineering.ru/ru/article/view?id=1278 (дата обращения: 01.05.2026).
2. Абдумиталип уулу К., Омаралиев, А. Ч., Арынова, К. А. &, Замирбек кызы Ы. (2026). Защита персональных и платёжных данных на серверах веб-приложений. Открытый журнал евразийских исследований, 1, 101-109. https://doi.org/10.65469/eijournal.2026.1.12

3. Ефимова, О.П., Ефимова, Н.А. (2020). Экономика общественного питания: учебное пособие. Москва: Инфра-М. С. 347.
4. Аркабаев, Н. & Алымова, З. (2024). Разработка Web серверных приложений на базе .NET CORE в примере интернет-магазина. Вестник Ошского государственного университета, 1, 142-154. https://doi.org/10.52754/16948610_2024_1_13
5. Chen, Z., Chai, N., Wang, J. & Wang, X. (2026). Restaurant recommendations under multimodal online reviews: A novel method based on image captioning and text analysis with multi-criteria decision-making. Information Processing & Management, 63(1), 104308. https://doi.org/10.1016/j.ipm.2025.104308
6. Медведев, М. А. (2024). Разработка веб-приложений на Django Framework : учебно-методическое пособие. Урал: Издательство Уральского университета. С. 148.
7. Мартин, Р. (2019). Чистая архитектура. Искусство разработки программного обеспечения. Санкт-Петербург: Питер. С. 418.
8. Дуйсебекова, К., Хабиров, Р. & Жолжан, А. (2021). Django как безопасный веб-фреймворк на практике. Вестник КазАТК, 116(1), 275-281. https://doi.org/10.52167/1609-1817-2021-116-1-275-281
9. Карабаев С.Э., Омаралиева Г.А., Исламбек кызы А., Жунусова Г.Б. (2026). Методы интеллектуального анализа и представления культурного наследия в распределенных веб-платформах (на примере Ошской области). Открытый журнал евразийских исследований, №1, сс. 131-143. doi: 10.65469/eijournal.2026.1.15
10. Методология тестирования безопасности веб-приложений на Django с акцентом на выявление уязвимостей бизнес-логики / А. Ч. Омаралиев, С. Э. Карабаев, Г. А. Омаралиева, В. Данг // Вестник Ошского государственного университета. – 2025. – № 4. – С. 199-211. – DOI 10.52754/16948610_2025_4_14. – EDN RNBTME.
11. Аркабаев, Н. К. Разработка web серверных приложений на базе.NET Core в примере интернет-магазина / Н. К. Аркабаев, З. Ж. Алымова // Вестник Ошского государственного университета. – 2024. – № 1. – С. 142-154. – DOI 10.52754/16948610_2024_1_13. – EDN GCXFII.

Евразия изилдөөлөрү ачык журналы, 2026, №2, бб. 116-127
doi: 10.65469/eijournal.2026.2.14
eijournal.ilimbilim.kg

ИНФОРМАТИКА / COMPUTER SCIENCE
УДК 004.04

Кофе-ресторандарын башкаруусун автоматташтыруу үчүн Django Framework негиздеги модулдук веб-тиркеме архитектурасы

Аркабаев Нуркасым Кылычбекович

ф.-м.и.к., доцент, Ош мамлекеттик университети, Кыргызстан, narkabaev@oshsu.kg,
ORCID: 0009-0000-1912-2225

Аскарова Лилия Сиюрбековна

магистрант, Ош мамлекеттик университети, Кыргызстан, askarovaliliyao8@gmail.com,
ORCID: 0009-0005-2060-4519

Айтматова Гулзада Досматовна

магистрант, Ош мамлекеттик университети, Кыргызстан, aitmatovao101@gmail.com,
ORCID: 0009-0008-6783-6431

Аннотация

Бул макалада Django Framework негизинде кофе ресторандарын башкарууну автоматташтыруу үчүн модулдук веб-тиркеме архитектурасын иштеп чыгуу жана ишке ашыруу маселеси каралат. Кыргызстандагы чакан жана орто бизнес үчүн алардын чектөөлөрүн аныктоо максатында, ресторандарды автоматташтыруунун учурдагы системаларынын (R-Keeper, iiko, Poster POS) талдоосу ишке ашырылы, алардын чектөөлөрү аныкталат. Ресторан системасы үчүн филиалдарды, персоналды, менюларды, кампаларды, буйрутмаларды жана аналитиканы башкаруу модулдарын камтыган алты модулдук архитектура сунушталат. Макалада системаны көз карандысыз Django тиркемелерине бөлүү принциптери, RESTful API аркылуу модулдардын ортосундагы өз ара аракеттенүү схемасы, маалымат моделдери жана ингредиенттерди автоматтык түрдө эсептен чыгаруу механизми баяндалат. Тестирлөөнүн жыйынтыктары системанын иштешин жана анын предметтик тармактын функционалдык талаптарына шайкештигин тастыктайт.

Ачкыч сөздөр: модулдук архитектура, веб-тиркеме, Django Framework, ресторан бизнесин автоматташтыруу, кофе ресторану, RESTful API, филиалдарды башкаруу

Open Journal of Eurasian Issues, 2026, no. 2, pp. 116-127
doi: 10.65469/eijournal.2026.2.14
eijournal.ilimbilim.kg

ИНФОРМАТИКА / COMPUTER SCIENCE
УДК 004.04

Modular Architecture of Web Applications for Coffee-Restaurant Management Automation Based on Django Framework

Nurkasym Kylychbekovich Arkabaev
Candidate of Phys-Math Sciences, Senior Lecturer, Osh State University, Kyrgyzstan, narkabaev@oshsu.kg, *ORCID: 0009-0000-1912-2225*

Liliya Siyurbekovna Askarova
Master's Student, Osh State University, Kyrgyzstan, askarovaliliyao8@gmail.com, *ORCID: 0009-0005-2060-4519*

Gulzada Dosmatovna Aitmatova
Master's Student, Osh State University, Kyrgyzstan, aitmatovao101@gmail.com, *ORCID: 0009-0008-6783-6431*

Abstract
This article discusses the design and implementation of a modular web application architecture for automating coffee-restaurant management based on the Django Framework. An analysis of existing restaurant automation systems (R-Keeper, iiko, Poster POS) was conducted, revealing their limitations for small and medium businesses in Kyrgyzstan. A six-module architecture for the "Restaurant" system is proposed, including modules for branch management, personnel, menu, inventory, orders, and analytics. The principles of system decomposition into independent Django applications, module interaction schemes via RESTful API, data models, and the automatic ingredient write-off mechanism are described. Testing results confirm the system's operability and compliance with domain functional requirements.

Keywords: modular architecture, web application, Django Framework, restaurant business automation, coffee-restaurant, RESTful API, branch management