



РАЗРАБОТКА МНОГОРОЛЕВОЙ ВЕБ-ПЛАТФОРМЫ ДЛЯ УПРАВЛЕНИЯ ПРОФЕССИОНАЛЬНОЙ ПРАКТИКОЙ СТУДЕНТОВ: АРХИТЕКТУРА, АЛГОРИТМЫ И ПРОГРАММНЫЕ РЕШЕНИЯ

Арзикулов Суннатулла Дониярович

Ташкентский государственный экономический университет

Старший преподаватель кафедры "Информационные системы и технологии"

E-mail: arzikulov.Sunnatulla@gmail.com

Аннотация: В статье представлены архитектура, основные алгоритмы и программные решения многоролевой веб-платформы, предназначенной для цифрового управления профессиональной практикой студентов. Исследование началось с выявления восьми недостатков традиционного, бумажного формата управления практикой в высших учебных заведениях Узбекистана. Предлагаемая платформа построена на трёхслойной архитектуре на базе технологий Django REST Framework и React.js и поддерживает 7 различных ролей пользователей (студент, HR-специалист предприятия, ментор, руководитель практики, кафедра, деканат, администратор). Для управления заявками разработан паттерн "Конечный автомат" (State Machine), включающий 9 состояний, а также алгоритм ежедневного двухэтапного подтверждения. База данных состоит из 12 таблиц и соответствует третьей нормальной форме (3NF). Пилотные испытания, проведённые в ТГЭУ, показали сокращение времени документооборота на 80% по сравнению с традиционной системой.

Ключевые Слова: студенческая практика, веб-платформа, Django REST Framework, React.js, аутентификация JWT, RBAC, паттерн "Конечный автомат", многоролевая система.

ВВЕДЕНИЕ

В современном информационном обществе задача обеспечения профессиональной подготовки студентов высшими учебными заведениями

приобретает всё большую значимость. В "Стратегии инновационного развития Республики Узбекистан на период до 2030 года" и стратегии "Цифровой Узбекистан — 2030" чётко определена необходимость цифровизации системы образования [1, 2]. В частности, 4-е направление стратегии "Узбекистан-2030" ставит задачу по цифровизации деятельности высших учебных заведений, включая автоматизацию процессов управления студенческой практикой.

В 2023-2024 учебном году число студентов в высших учебных заведениях Узбекистана превысило 1,2 миллиона человек. Более 35% из них ежегодно проходят профессиональную практику на предприятиях и в организациях. Однако управление этим процессом до сих пор ведётся в традиционной бумажной форме: заполняемые вручную таблицы учёта посещаемости, направления, отправляемые по почте, и отчёты, с которых снимаются копии. В ходе исследования были проанализированы пять этапов этого процесса, и на каждом из них выявлены серьёзные недостатки: потеря документов, сложности в координации, отсутствие мониторинга в реальном времени и разрыв связи между тремя сторонами (университетом, предприятием и студентом) [3].

Закон Республики Узбекистан "О высшем образовании и научно-исследовательской деятельности" (2022) предусматривает оформление договоров о прохождении практики между высшими образовательными учреждениями и предприятиями в электронной форме. Методическое указание Министерства высшего и среднего специального образования от 2021 года устанавливает порядок и минимальные требования к документированию студенческой практики [4]. Однако, несмотря на то, что оформление документов предусмотрено в электронном виде, специальной цифровой платформы для управления ими не существует.

В ходе исследования были проведены интервью с сотрудниками кафедры ИКТ Ташкентского государственного экономического университета и партнерских предприятий. Результаты показали следующее: сотрудники

кафедры тратят на организацию практики в среднем 6-8 часов в неделю; сотрудники деканата вручную подготавливают 150-200 направлений за семестр; наставники на предприятиях не могут контролировать заполнение дневников; студенты не знают, на каком этапе находится рассмотрение их заявления.

Существующие на сегодняшний день зарубежные платформы — Handshake, Symplicity, GradLeaders, Moodle — не могут удовлетворить специфические требования системы образования Узбекистана: отсутствует поддержка местного языка, нет адаптации к национальному законодательству, а цены слишком высоки (15 000–30 000 долларов США в год). В свою очередь, местная система HEMIS не включает в себя модуль управления практикой [4]. По этой причине создание специальной цифровой платформы для вузов Узбекистана, полностью отвечающей местным требованиям, стало научно-практической необходимостью.

Цель исследования — разработка и представление архитектуры, алгоритмов и программных решений веб-платформы, построенной на многоуровневой основе, для управления профессиональной практикой студентов. Объект исследования — процесс организации и управления профессиональной практикой студентов в высших учебных заведениях. Предмет исследования — архитектура и алгоритмы веб-платформы, предназначенной для автоматизации данного процесса.

Научная новизна работы заключается в том, что впервые разработана веб-платформа, адаптированная к системе образования Узбекистана, поддерживающая 7 различных ролей пользователей и управляющая жизненным циклом заявки на основе паттерна "Конечный автомат" (State Machine) с 9 состояниями. Практическая значимость заключается в том, что платформа была успешно апробирована кафедрой АТТ ТГЭУ в весеннем семестре 2026 года.

ОБЗОР ЛИТЕРАТУРЫ И АНАЛИЗ ПРОБЛЕМЫ

Международные исследования, посвящённые системам управления студенческой практикой, показывают наличие в этой сфере серьёзного пробела. Согласно отчёту NACE (Национальной ассоциации колледжей и работодателей) за 2022 год, в США 73% предприятий и университетов предпочитают цифровое управление процессами стажировок, однако лишь 41% учреждений располагают такими возможностями [5]. Бриз и Фуртадо (2021) проанализировали 47 исследований по автоматизации административных процессов в университетах и установили, что цифровые системы могут сократить время документооборота в среднем на 68% [6]. Этот показатель усиливает научное обоснование для создания новой платформы.

В вузах Узбекистана традиционный процесс управления практикой состоит из пяти этапов: (1) определение места практики — заведующий кафедрой составляет список предприятий (2–3 недели); (2) оформление документов — студент получает направление в деканате (3–5 дней); (3) начало практики — направление передаётся на предприятие; (4) прохождение практики — табель посещаемости и ежедневный журнал, подписываемые вручную; (5) отчёт и завершение — письменный отчёт (20–40 страниц) и сдача документов. В ходе исследования на каждом из этих пяти этапов было выявлено восемь системных проблем: утеря документов, задержки, отсутствие мониторинга в реальном времени, риск фальсификации, избыточный расход бумаги, транспортные расходы, проблемы с координацией и отсутствие статистики.

Анализ зарубежных платформ: Handshake (США, 2014, 1400+ вузов) — в основном ориентирована на карьерный центр, отсутствуют модули учёта посещаемости и ежедневного контроля в период практики, стоимость составляет 8 000–25 000 долларов в год, местная локализация отсутствует. Symplicity (1996) — функционально полноценная, однако её стоимость очень высока и она не соответствует требованиям законодательства Узбекистана.

GradLeaders — специализируется на взаимоотношениях "вуз-предприятие," однако мониторинг в период практики слабый. Moodle — широко распространённая в мире LMS, но не предназначенная для управления практикой. В местной же системе HEMIS отсутствуют модули подачи заявок, интерфейс предприятия, назначения ментора и учёта посещаемости [4, 7].

В результате анализа пробелов (Gap Analysis) было определено, что новая платформа должна соответствовать следующим пяти основным требованиям: (1) система контроля доступа на основе ролей (RBAC) с 7 ролями; (2) многоступенчатая автоматизированная цепочка утверждения заявок; (3) мониторинг посещаемости, задач и ежедневной отчётности в реальном времени; (4) алгоритм расчёта взвешенной итоговой оценки; (5) полная адаптация к узбекскому языку, законодательству Узбекистана и структуре вузов.

Международные исследования эффективности цифровизации: Согласно отчету ОЭСР "Взгляд на образование 2023" (Education at a Glance 2023), университеты, которые цифровизировали административное управление, сократили свои административные расходы в среднем на 23%. Отчет МОТ (Международная организация труда) за 2022 год показал, что в университетах с цифровой системой управления показатель трудоустройства по итогам практики на 34% выше.

В контексте Узбекистана Холматов и Турсунов (2023) проанализировали актуальные вопросы цифровизации образования и указали, что цифровизация управления практикой является одним из приоритетных направлений [19]. В свою очередь, исследование Каримова (2022) методологически обосновало оптимальность RESTful-архитектуры для платформы по организации практики [20].

Был проведён сравнительный анализ для выбора технологий. Django REST Framework был выбран как самый популярный веб-фреймворк на Python согласно опросу Stack Overflow 2023; LTS-версия (Django 4.2, до 2026 года)

гарантирует надёжность для производственной среды. React.js — благодаря таким преимуществам, как виртуальный DOM и компонентная архитектура, имеет более 15 миллионов npm-загрузок в месяц. PostgreSQL — полная совместимость с Django ORM и мощный оптимизатор для сложных запросов. JWT (стандарт RFC 7519) — аутентификация без сохранения состояния (stateless) снижает нагрузку на сервер [8]. Паттерн State Machine (Гамма и др., 1994) был применён для управления жизненным циклом заявки [9]. Принципы Enterprise Application Architecture Фаулера (2002) легли в основу проектирования слоистой архитектуры [10].

Модуль практики Moodle используется в некоторых вузах Узбекистана, так как он доступен и бесплатен. Однако у него есть серьёзные ограничения: отсутствует интерфейс для предприятия, нет цепочки утверждения заявок, нет контроля посещаемости, отсутствует полноценная поддержка местного языка. Для настройки требуется специальный IT-специалист, а мобильный интерфейс недостаточно развит. Вывод: Moodle подходит для управления общим образовательным процессом, но в нём отсутствуют ключевые функции для управления профессиональной практикой.

Результаты сравнительного анализа. Сравнительный анализ, проведённый по десяти критериям (многоуровневый доступ, адаптация к местному языку, цепочка заявлений, контроль посещаемости, система электронного дневника, QR-код, расчёт оценок, соответствие законодательству, цена и техническая поддержка), показал, что ни одна из существующих систем не удовлетворяет всем критериям одновременно. Разработанная платформа, как единое решение, получила положительную оценку по всем 10 критериям.

МЕТОДОЛОГИЯ

При разработке платформы применялся вариант Scrum методологии Agile. Работа была разделена на 2-недельные спринты: Спринт 1 — анализ

требований и модель базы данных; Спринт 2 — аутентификация JWT и RBAC; Спринт 3 — управление заявлениями (State Machine); Спринт 4 — модули электронного дневника и посещаемости; Спринт 5 — дашборды фронтенда; Спринт 6 — система QR-кодов и направлений; Спринт 7 — UAT и оптимизация. В конце каждого спринта предоставлялся работоспособный фрагмент продукта.

Проектирование было основано на принципах предметно-ориентированного проектирования (Domain-Driven Design, DDD). Система организована в пяти основных доменах: Пользователи, Операции, Дневники, Документы, Оценка. Каждый домен реализован как независимое приложение Django. UML-диаграммы — диаграммы прецедентов, классов, последовательности и состояний — послужили основным инструментом при системном анализе и проектировании. ER-диаграмма была использована для проектирования структуры базы данных.

При выборе технологий применялись следующие критерии: (1) лицензия с открытым исходным кодом — лицензии Django (MIT), React (MIT), PostgreSQL (BSD) сводят финансовые затраты к нулю; (2) совместимость с существующей серверной инфраструктурой в университетах Узбекистана; (3) наличие LTS-версии — долгосрочная поддержка в производственной среде; (4) широта экосистемы и качество документации. Для бэкенда также рассматривались FastAPI и Node.js, однако встроенная панель администратора, система ORM и обширная экосистема Django обеспечили ему преимущество.

В исследовании применялся экспериментальный метод: создание прототипа и его тестирование с реальными пользователями в университете ТГЭУ. Метод сравнительного анализа использовался для сопоставительной оценки с зарубежными платформами и традиционной системой. При сборе данных использовались методы наблюдения и анкетирования.

Совокупность функциональных (FR) и нефункциональных (NFR) требований была сформирована на основе интервью с заинтересованными сторонами и анализа существующих систем. Количество FR: 34 (аутентификация — 4, управление практикой — 12, дневник — 8, документы — 5, оценивание — 5). Количество NFR: 12 (производительность, безопасность, масштабируемость, удобство использования и локализация). Был применен метод приоритизации MoSCoW: Must Have (18 FR), Should Have (10 FR), Could Have (6 FR).

Для сбора данных использовались три метода. Первый — анализ документов: были изучены существующие бумажные направления, таблицы посещаемости и шаблоны отчётов. Второй — наблюдение: реальное состояние процесса отслеживалось в течение двух недель совместной работы с сотрудниками кафедры АТТ. Третий — опрос: были выявлены потребности в цифровой системе среди 50 студентов и 15 сотрудников. Комбинация этих трёх методов позволила полностью сформировать требования.

АРХИТЕКТУРА СИСТЕМЫ И БАЗА ДАННЫХ

Трёхслойная архитектура. Платформа построена на основе классической трёхслойной (3-tier) архитектуры. Первый слой — Уровень представления (Presentation Layer): одностраничное приложение (Single Page Application, SPA) на базе React.js 18.2. Второй слой — Уровень бизнес-логики (Business Logic Layer): Django 4.2 и Django REST Framework 3.14. Третий слой — Уровень данных (Data Layer): реляционная база данных PostgreSQL 15. Трёхслойная архитектура позволяет независимо изменять, тестировать и масштабировать каждый слой. Связь между слоями осуществляется через RESTful API (в формате JSON). При выборе архитектуры сравнивались монолитный и микросервисный подходы: для данного объёма проекта и размера команды монолитная архитектура была признана более оптимальной,

но при этом спроектирована с учётом готовности к будущему переходу на микросервисы.

Обоснование выбора монолитной архитектуры. На этапе проектирования сравнивались монолитный и микросервисный архитектурные подходы. Преимущества монолита: простота развёртывания, единая кодовая база, лёгкость управления транзакциями, удобство для команды. Преимущества микросервисов: независимое масштабирование, свобода в выборе технологий. С учётом масштаба проекта и размера команды была выбрана монолитная архитектура. Однако, поскольку система основана на API, реализованном через Django REST Framework, в будущем будет легко перейти на микросервисы: каждое приложение Django можно будет преобразовать в отдельный микросервис.

Модель базы данных. База данных состоит из 12 таблиц: User, UserProfile, Company, Internship, Application, DailyLog, Attestation, Task, Document, Grade, Notification, AuditLog. При проектировании была выполнена нормализация до третьей нормальной формы: 1NF — все атрибуты атомарны; 2NF — отсутствуют частичные зависимости; 3NF — отсутствуют транзитивные зависимости. Центральным объектом системы является таблица Internship (Стажировка), которая связана с таблицами User (студент), Company (предприятие), Application (заявка), DailyLog (дневник) и Document (документ). Каскадное удаление (CASCADE) применяется только для записей DailyLog; в остальных связях используется PROTECT, что предотвращает случайную потерю данных.

Модель пользователя и RBAC. В системе вместо стандартной модели Django User используется кастомная модель на основе AbstractUser. Это позволяет реализовать аутентификацию по email и добавить поле для роли (role). Матрица RBAC включает 161 запись о разрешениях для 7 ролей × 23 ресурсов. К каждой конечной точке API (API endpoint) привязаны классы DRF IsAuthenticated и специальный класс InternshipRolePermission. Такой подход

позволяет избежать прямых проверок ролей в коде и обеспечивает централизованное управление разрешениями.

Стратегия индексирования. Для ускорения запросов к базе данных были добавлены индексы к столбцам, по которым часто выполняется фильтрация (status, user_id, date, assignment_id). Даже при наличии более 1000 записей в DailyLog запросы с фильтрацией занимали менее 10 мс. Методы select_related () и prefetch_related () устранили проблему N+1, значительно сократив количество запросов.

Миграция данных и резервное копирование. Фреймворк миграций Django управляет каждым изменением схемы, обеспечивая версионирование. Требовалось, чтобы каждая миграция была обратимой (reversible). Резервное копирование выполняется автоматически 3 раза в день; доступна возможность восстановления на определённый момент времени (point-in-time recovery) на основе PostgreSQL WAL (Write-Ahead Logging). В течение тестового периода система поддерживала показатель безотказной работы (uptime) на уровне 99,7%.

Многоуровневая безопасность. Система безопасности состоит из четырёх уровней: HTTPS/TLS на транспортном уровне; stateless-токены JWT на уровне аутентификации; классы разрешений RBAC на уровне авторизации; изоляция на уровне строк (row-level) через переопределение get_queryset () на уровне данных. Настройки CORS ограничивают приём запросов только от разрешённых источников (origin). Все конфиденциальные данные управляются через переменные окружения (environment variables).

ОСНОВНЫЕ АЛГОРИТМЫ

Управление заявками: конечный автомат (State Machine) с 9 состояниями. Жизненный цикл заявки студента реализован на основе паттерна "Конечный автомат" (State Machine). Заявка может находиться в одном из 9 состояний: DRAFT (черновик) → SUBMITTED (подана) →

UNDER_REVIEW (на рассмотрении) → APPROVED_BY_DEPARTMENT (одобрена кафедрой) → APPROVED_BY_DEAN (одобрена деканатом) → PLACEMENT_CONFIRMED (подтверждена предприятием) → ACTIVE (активная практика) → COMPLETED (завершена) / REJECTED (отклонена).
Определены 7 разрешённых переходов и 4 состояния для отката (rollback).

Основная причина выбора паттерна "Конечный автомат" (State Machine) — предотвращение недопустимых переходов между состояниями. Например, заявление, не утверждённое кафедрой, не может быть напрямую утверждено деканатом. Для каждой пары "состояние-переход" определены: а) класс разрешений (permission), проверяющий права доступа; б) обертывание в атомарную транзакцию (atomic transaction); в) запись в журнал аудита; г) триггер уведомлений. При попытке недопустимого перехода на уровне API возвращается ошибка HTTP 422 (Unprocessable Entity). Журнал аудита для каждого перехода фиксирует предыдущее состояние, новое состояние, временную метку и идентификатор (ID) ответственного пользователя.

Алгоритм двухэтапного подтверждения ежедневной записи. Для подтверждения записи в ежедневнике (DailyLog) внедрён двухэтапный алгоритм. Первый этап — подтверждение ментором (представителем предприятия): после того как студент вносит описание работы и количество отработанных часов, ментор рассматривает и подтверждает запись либо возвращает её с замечаниями. Второй этап — подтверждение руководителем практики (представителем вуза): этот этап возможен только после успешного завершения первого. Основное преимущество данного алгоритма заключается в том, что контроль распределён между двумя независимыми сторонами (предприятием и вузом): если одна сторона подделает данные, вторая их проверит и заблокирует. Утверждение руководителем практики будет заблокировано на уровне API, если не выполнено условие `approved_by_mentor = True`.

Алгоритм для направлений и учёта посещаемости по QR-коду.

Система направлений (instruction letter) переводит официальный документ, принятый в системе высшего образования Узбекистана, в цифровой формат. При создании направления генерируется уникальный QR-код, подписанный с помощью HMAC-SHA256, который встраивается в PDF-документ посредством библиотеки ReportLab и рассылается по учебной группе с электронной подписью представителя деканата. QR-код также применяется в системе учёта посещаемости: система ежедневно генерирует для организации уникальный QR-код, действительный в течение 24 часов; студент по прибытии сканирует его через мобильный браузер; сервер проверяет подпись и фиксирует время сканирования и GPS-координаты.

Для технической реализации системы QR-кодов были использованы Python-библиотека qrcode и графическая библиотека Pillow (PIL). Каждый сгенерированный код сохраняется в базе данных в виде хеш-значения, что повышает безопасность, так как позволяет избежать хранения изображений на сервере. Для сканирования из мобильного браузера использовалась JavaScript-библиотека html5-qrcode, что избавляет от необходимости устанавливать отдельное мобильное приложение. Данное архитектурное решение сводит к минимуму барьеры входа для пользователя.

Алгоритм расчёта итоговой оценки. По завершении практики итоговая оценка студента рассчитывается из двух компонентов: оценки ментора (вес 60%) и оценки руководителя практики (вес 40%). Если посещаемость студента составляет менее 80%, итоговая оценка автоматически снижается на один балл. Шкала оценок: 86–100 = "Отлично" (5), 71–85 = "Хорошо" (4), 56–70 = "Удовлетворительно" (3), 55 и ниже = "Неудовлетворительно" (2). Алгоритм запускается автоматически, когда практика переходит в статус COMPLETED, посредством фреймворка сигналов Django, и уведомляет все заинтересованные стороны по электронной почте.

Для реализации конечного автомата (State Machine) был создан специальный класс-миксин `InternshipStateMachineMixin`. Этот миксин предоставляет метод `transition()`, который: проверяет допустимость входящего перехода состояния по разрешённой матрице, возвращает HTTP 422 при некорректных переходах, записывает лог аудита при успешном переходе и отправляет уведомление по электронной почте. Миксин обеспечивает единую логику для всех 9 состояний, избавляя от дублирования кода.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Архитектура бэкенда. На стороне бэкенда было разработано 45 API-эндпоинтов на базе Django REST Framework 3.14. Эндпоинты разделены на четыре группы: аутентификация и пользователи (8), стажировки и заявки (14), дневники и задачи (12), документы и оценка (11). Структура проекта разделена на 5 приложений (apps) Django: `users`, `internships`, `daily_logs`, `documents`, `grades`. Каждое приложение состоит из независимых файлов: `models.py`, `serializers.py`, `views.py` и `permissions.py`. Классы `ViewSet` и `Serializer` из DRF спроектированы в соответствии с принципами SOLID: каждый класс несет единственную ответственность.

Аутентификация и безопасность. Аутентификация на основе JWT (JSON Web Token) реализована с помощью библиотеки `django-rest-framework-simplejwt`. Access-токен действителен в течение 15 минут, refresh-токен — 7 дней. При каждом запросе access-токен передается в заголовке `Authorization` по схеме `Bearer`. Безопасность обеспечена в соответствии с рекомендациями OWASP Top 10: защита от SQL-инъекций — посредством параметризованных запросов Django ORM; от XSS — с помощью библиотеки `DOMPurify` и заголовков `Content Security Policy`; от CSRF — через middleware Django; ограничение частоты запросов (rate limiting) — установлено на 20 запросов в минуту с одного IP-адреса через DRF Throttling; все пароли хешированы с использованием алгоритма `bcrypt`.

Фильтрация на основе ролей (RBAC). В каждом ViewSet переопределён метод `get_queryset()`, который возвращает данные в строгом соответствии с ролью пользователя. Студент видит только свои заявки, наставник — только своих студентов, а кафедра — только практики своей кафедры. Такой подход обеспечивает изоляцию данных, блокируя несанкционированный доступ к информации других пользователей на уровне API. Для версионирования API была выбрана стратегия на основе URL-путей (`/api/v1/...`), чтобы при будущем переходе на v2 не нарушить существующие интеграции.

Структура системы. Архитектура бэкенд-проекта разделена на 5 независимых Django-приложений: `users` (модель пользователя, JWT, профиль), `internships` (практика, заявка, конечный автомат), `daily_logs` (ежедневный отчёт, посещаемость, двухэтапное подтверждение), `documents` (направление, QR-код, генерация PDF), `grades` (оценка, аттестация). Каждое приложение состоит из самостоятельных файлов: `models.py`, `serializers.py`, `views.py`, `urls.py` и `permissions.py`. Фреймворк сигналов Django реализует событийно-ориентированные элементы: при изменении статуса заявки автоматически отправляется email-уведомление.

DRF ViewSet и Serializer. С помощью класса `ModelViewSet` реализованы стандартные CRUD-операции (`list`, `retrieve`, `create`, `update`, `destroy`) с минимальным количеством кода. Специальный декоратор `@action` был использован для операций переходов конечного автомата (State Machine), таких как `submit`, `approve`, `reject`. Вложенные сериализаторы (Nested serializers) возвращают связанные данные в одном API-запросе: например, профиль студента, название предприятия и текущий статус передаются вместе с данными о практике в едином JSON-ответе. Через `SerializerMethodField` вычисляются и возвращаются динамические поля, такие как процент посещаемости и количество утвержденных дневников.

Фронтенд и пользовательский интерфейс. На основе React.js 18.2 создано 7 отдельных ролевых панелей (dashboard). Управление состоянием реализовано посредством комбинации Context API и useReducer: AuthContext (сессия пользователя), InternshipContext (данные о практике). Для маршрутизации используется React Router DOM v6; для каждой роли предусмотрен отдельный префикс URL: /student/*, /mentor/*, /supervisor/*, /department/*, /dean/*, /admin/*. Библиотека Axios управляет HTTP-взаимодействием: перехватчики (interceptors) автоматически добавляют JWT-токен и обновляют его с помощью refresh-токена при получении ошибки с кодом 401. Платформа разработана в соответствии со стандартами Progressive Web App (PWA), что дает студентам возможность установить ее на мобильное устройство.

Панель студента. На главной странице (дашборде) студента отображаются четыре статистические карточки: количество утвержденных дневников, ожидающие подтверждения (не утвержденные ментором), количество выполненных заданий и процент посещаемости. На странице заявок студент может отправлять отклики на вакансии и отслеживать текущий статус своих заявок в виде диаграммы состояний (State Machine). На странице дневника студент ежедневно вводит описание проделанной работы, количество отработанных часов и может в реальном времени видеть статус подтверждения от ментора и супервайзера.

Панели деканата и кафедры. Панель деканата обладает большинством административных функций системы: создание направлений по группам и их экспорт в PDF, управление договорами с предприятиями, общий отчет по всем практикам, статистические дашборды (с графиками и диаграммами). Панель кафедры позволяет заведующему отслеживать студентов своей кафедры и быстро определять, какие студенты активны, а у каких есть проблемы. Все панели имеют функцию экспорта (в Excel, PDF).

Описание ролевых панелей. Панель студента: статус практики, статистические карточки (подтверждённые дневники, ожидающие и выполненные задания, процент посещаемости), подача заявления, ведение дневника. Панель ментора: список своих студентов, система подтверждения дневников, постановка задач. Панель руководителя практики: система второго этапа подтверждения дневников, утверждённых ментором, выставление итоговой оценки. Панель кафедры: мониторинг студентов кафедры, общая статистика. Панель деканата: создание направлений по группам и экспорт в PDF, управление договорами с предприятиями, общий отчёт по всем практикам. Панель администратора: управление пользователями и предприятиями, настройки системы.

РЕЗУЛЬТАТЫ И ОБСУЖДЕНИЕ

Кафедрой АТТ ТГЭУ были проведены пилотные испытания в весеннем семестре 2026 года. Участники: 15 студентов, 5 менторов, 4 руководителя практики, 2 сотрудника кафедры, 2 сотрудника деканата (всего 28 человек). Испытания проводились в течение 8 недель в процессе реальной практики.

Результаты сравнительного анализа с традиционной системой: время документооборота сократилось с 12 до 2,3 дня (–80,8%); случаи утери документов были полностью устранены (в предыдущем семестре было зафиксировано 3 таких случая); возможность фальсификации контроля посещаемости была исключена благодаря системе QR-кодов; благодаря мониторингу в реальном времени кафедра и деканат ежедневно отслеживали ход практики.

Сравнение с существующими платформами: Handshake — отсутствует узбекский язык и поддержка законодательства, цена \$25 000/год; Symplicity — отсутствует локализация, \$30 000/год; HEMIS — отсутствует модуль управления практикой; разработанная платформа — полная поддержка узбекского языка и законодательства Узбекистана, бесплатная (открытый

исходный код). При этом она является единственным местным решением, удовлетворяющим всем 8 критическим требованиям.

Ключевые преимущества архитектуры платформы: во-первых, паттерн State Machine позволил полностью формализовать процесс обработки заявок и заблокировал любые некорректные переходы состояний — в ходе тестирования не возникло ни одного неверного состояния. Во-вторых, двухэтапный алгоритм ежедневного подтверждения распределил контроль между двумя независимыми сторонами, исключив возможность ложных подтверждений. В-третьих, система RBAC обеспечила работу каждого пользователя строго в рамках своих функций.

Также учтены ограничения платформы: в текущей версии отсутствует отдельное мобильное приложение (только PWA); система подбора студента к предприятию на основе ИИ запланирована на будущее; на данный момент платформа сконфигурирована только для одного университета (требуется переход на многопользовательскую архитектуру). Эти ограничения определяют направления для дальнейших исследований.

Анализ производительности (Apache JMeter): при параллельной работе 100 виртуальных пользователей в течение 10 минут среднее время отклика составило 142 мс; 95-й перцентиль — 287 мс; уровень ошибок — 0,28%; максимальная пропускная способность — 4 847 запросов в минуту. Методы `select_related()` и `prefetch_related()` устранили проблему "N+1," сократив количество запросов к базе данных на 70%. Кэширование с помощью Redis уменьшило время отклика для часто запрашиваемых эндпоинтов до 8 мс.

Удобство пользовательского интерфейса оценивалось по методологии SUS (System Usability Scale) 28 реальными пользователями (Brooke, 1996) [18]. Общий балл по шкале SUS составил 82,3, что соответствует уровню "Отлично" и значительно превышает средний показатель по отрасли образовательных систем (72,5). Наивысший балл: точность навигации на основе ролей (4.7/5.0).

Наименьший балл: процесс первоначальной регистрации (3.9/5.0) — в будущем планируется добавить мастер настройки (onboarding wizard).

Экономический анализ показывает, что внедрение платформы обеспечит университету значительную ежегодную экономию ресурсов: сократятся затраты рабочего времени сотрудников на подготовку и согласование документов; будут устранены расходы на бумагу, печать и почтовые отправления. Наличие цифрового архива создаст ценную информационную базу для будущего анализа данных о процессе прохождения практики и повышения её качества.

ЗАКЛЮЧЕНИЕ

В данной статье представлена архитектура, основные алгоритмы и программные решения многоролевой веб-платформы, разработанной для цифрового управления профессиональной практикой студентов. Были получены следующие основные результаты.

1. Выявлены восемь системных недостатков в организации традиционной практики в вузах Узбекистана. Проанализированы четыре зарубежные платформы (Handshake, Symplicity, GradLeaders, Moodle) и отечественная система HEMIS по десяти критериям — доказано, что ни одна из них не удовлетворяет в полной мере местным требованиям.

2. Построена трёхслойная архитектура на основе технологий Django REST Framework, React.js 18.2 и PostgreSQL 15. База данных из 12 таблиц соответствует требованиям третьей нормальной формы (3NF). Матрица RBAC управляет 161 записью разрешений для схемы "7 ролей × 23 ресурса."

3. Внедрены два основных алгоритма: шаблон "конечный автомат" (State Machine) с 9 состояниями для управления заявками (7 разрешённых переходов) и двухэтапный алгоритм утверждения для дневника практики. Реализованы алгоритм генерации QR-кода с подписью HMAC-SHA256 для создания направления и алгоритм расчёта взвешенной итоговой оценки.

4. На стороне backend обеспечены 45 конечных точек RESTful API, аутентификация по токену JWT, фильтрация по ролям RBAC и безопасность в соответствии с рекомендациями OWASP Top 10. На стороне frontend реализованы 7 ролевых панелей, поддержка PWA и перехватчики Axios.

Пилотные испытания, проведённые в ТГЭУ, подтвердили, что время документооборота сократилось на 80,8% по сравнению с традиционной системой. Случаи утери документов были полностью устранены.

Научная значимость работы заключается в том, что впервые была разработана и апробирована на практике модель State Machine с 9 состояниями и многоролевая матрица RBAC для управления студенческой практикой, адаптированная к системе образования Узбекистана. Направления для дальнейшего развития: система адаптации студента к предприятию на основе искусственного интеллекта, отдельное мобильное приложение и переход на многопользовательскую (multi-tenant) архитектуру для других вузов.

Экономический эффект от внедрения платформы: сократилось рабочее время сотрудников, затрачиваемое на подготовку и согласование документов; были устранены расходы на бумагу, печать и почтовые услуги. Наличие цифрового архива создаёт ценную базу для анализа данных о процессе прохождения практики и повышения её качества в будущем: какие предприятия принимают больше всего студентов, какая кафедра наиболее эффективно организует практику.

Теоретический вклад: (1) предложена и апробирована модель State Machine с 9 состояниями для управления практикой, адаптированная к среде высшего образования Узбекистана; (2) модель многоролевой матрицы RBAC впервые протестирована в местном контексте; (3) двухэтапный алгоритм контроля посещаемости впервые применён в образовательной практике Узбекистана. Данные результаты могут быть применены и в других вузах Узбекистана.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ (REFERENCES)

1. Указ Президента Республики Узбекистан "О Стратегии инновационного развития Республики Узбекистан на период до 2030 года." — Ташкент, 2020.
2. Стратегия "Цифровой Узбекистан — 2030." — Ташкент: Администрация Президента Республики Узбекистан, 2020. — 45 с.
3. Закон Республики Узбекистан "О высшем образовании и научно-исследовательской деятельности" (новая редакция). — Ташкент, 2022. — 48 с.
4. Министерство высшего и среднего специального образования. Методическое указание по организации профессиональной практики студентов. — Ташкент, 2021. — 32 с.
5. NACE (National Association of Colleges and Employers). Internship & Co-op Survey Report. — 2022. — 58 p.
6. Breese R., Furtado A. Automating university administrative processes: a systematic review // Journal of Higher Education Policy and Management. — 2021. — Vol. 43. — P. 112–128.
7. Официальная документация системы HEMIS. Информационная система управления высшим образованием. — hemis.uz, 2023.
8. JSON Web Tokens. RFC 7519. — Internet Engineering Task Force (IETF), 2015.
9. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. — Addison-Wesley, 1994. — 395 p.
10. Fowler M. Patterns of Enterprise Application Architecture. — Addison-Wesley, 2002. — 560 p.
11. Holovaty A., Kaplan-Moss J. The Definitive Guide to Django. 2nd ed. — Apress, 2009. — 536 p.
12. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2-е изд. — O'Reilly Media, 2020. — 310 с.

13. Richardson L., Ruby S. RESTful Web Services. — O'Reilly Media, 2007. — 448 с.
14. Evans E. Domain-Driven Design: Tackling Complexity in the Heart of Software. — Addison-Wesley, 2003. — 560 с.
15. OWASP Foundation. OWASP Top Ten 2023. — owasp.org, 2023.
16. Kleppmann M. Designing Data-Intensive Applications. — O'Reilly Media, 2017. — 616 с.
17. PostgreSQL Global Development Group. PostgreSQL 15 Documentation. — postgresql.org, 2023.
18. Django REST Framework. Official Documentation v3.14. — django-rest-framework.org, 2023.
19. Холматов А., Турсунов Б. Актуальные вопросы цифровизации образования в Узбекистане // Журнал информационных технологий. — 2023. — No 4. — С. 12–19.
20. Каримов Д. Сравнение подходов REST и GraphQL // Научный вестник ТГЭУ. — 2022. — No 2. — С. 45–52.