Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и управление» (ИУК)

КАФЕДРА «Системы автоматического управления и электротехника» (ИУК3)

О Т Ч Е Т

УЧЕБНАЯ ПРАКТИКА

«Ознакомительная практика»

Выполнил: студент гр. ИУК3-41Б _____ _____
(подпись)

Проверил: Финошин А.В._____
(подпись)

Оценка руководителя _____ баллов_____
30-50 (дата)

Оценка защиты _____ баллов _____
30-50 (дата)

Оценка практики _____ баллов _____
(оценка по пятибальной шкале)

Комиссия: _____(_____)
(подпись) (Ф.И.О.)

_____(_____)
(подпись) (Ф.И.О.)

_____(_____)
(подпись) (Ф.И.О.)

Калуга, 2025

Министерство науки и высшего образования Российской Федерации Калужский филиал федерального государственного автономного образовательного учреждения высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой __ИУК3___
_____(_____)
«____»__2025 г.

З А Д А Н И Е
на УЧЕБНУЮ, ОЗНАКОМИТЕЛЬНУЮ ПРАКТИКУ

За время прохождения практики студенту необходимо:
1. Собрать основные сведения о предприятии–базе практики.
2. Подготовить окружение: Настроить Go (1.20+), PostgreSQL 13+, golang-migrate, docker compose. Создать .env с переменными (DB_URL, JWT_SECRET и др.).
3. Реализовать базовый API: Подключить Gin/Echo, настроить JWT-аутентификацию, описать модели и миграции для таблиц: users, tickets, comments, attachments и др. Создать основные CRUD-эндпоинты.
4. Добавить бизнес-логику и аудит: Обработать роли, права, soft-delete, историю изменений, полнотекстовый поиск, триггеры updated_at, DDL-аудит. Настроить индексирование и партиционирование.
5. Проверить работоспособность: Написать unit- и integration-тесты (покрытие ≥80%), задокументировать API через Swagger, убедиться, что миграции работают «с нуля», и всё запускается через docker-compose.
6. Проанализировать результат работы, подготовить отчёт о проделанной работе.

Дата выдачи задания « 30 » июня 2025 г.

Руководитель практики _____(_____)
(подпись) (Ф.И.О.)

Задание получил студент гр. ИУК3- 41Б _____(_____)
(подпись) (Ф.И.О.)

Содержание

ВВЕДЕНИЕ

В рамках прохождения практики была поставлена задача по разработке клиент-серверного модуля управления тикетами. Выполнение данного проекта позволило сформировать, закрепить и развить практические навыки и компетенции в области контроля, диагностики, системного анализа и управления, что является основной целью практики. Для достижения поставленной цели и успешной реализации индивидуального задания в ходе практики решались следующие задачи:

- Закрепление, систематизация и углубление теоретических знаний, полученных при освоении дисциплин образовательной программы, применительно к разработке современных программных систем.

- Формирование компетенций, связанных с будущей профессиональной деятельностью, через разработку архитектуры, бэкенда и фронтенда системы управления тикетами.

- Получение опыта и навыков практической деятельности, связанных с будущей профессиональной деятельностью, включая работу с базами данных, сетевым взаимодействием, аутентификацией и развертыванием приложений.

1. Основная часть

1.1. Изучение структуры предприятия – базы практики

Предприятием – базой прохождения практики является ФИЛИАЛ АО "НПО ЛАВОЧКИНА" В Г. КАЛУГА, который является частью крупной ракетно-космической промышленности России, АО "Научно-производственное объединение имени С.А. Лавочкина".

- Историческая справка

  Основное акционерное общество "НПО им. С.А. Лавочкина" является ведущей организацией в ракетно-космической отрасли, занимающейся разработкой, изготовлением и использованием автоматических космических комплексов и систем для фундаментальных научных исследований, астрофизики, планетарных исследований, дистанционного зондирования Земли, а также разгонных блоков. Калужский филиал АО "НПО Лавочкина" по адресу: Калужская обл., г. Калуга, ул. Октябрьская, д. 17.

- Основные виды деятельности и продукция:

  Основной вид деятельности АО "НПО Лавочкина" (включая филиалы) – это научные исследования и разработки в области естественных и технических наук. Также указаны виды деятельности, связанные с разработкой, производством, испытанием и ремонтом авиационной техники.

- Совершенствование и модернизация:

  Деятельность предприятия в сфере научных исследований и разработок подразумевает постоянное совершенствование, модернизацию и унификацию выпускаемых изделий. Внедрение систем документооборота и автоматизация рабочих мест также являются частью усилий по оптимизации процессов и повышению эффективности.

1.2. Научно-исследовательский этап

На данном этапе практики был проведен комплекс научно-исследовательских работ, ориентированных на теоретическое обоснование и прикладное проектирование системы управления тикетами. Основной задачей являлось не только изучение функциональных требований, но и глубокий анализ технологических решений, способных обеспечить высокую производительность, безопасность и масштабируемость разрабатываемого модуля. Это позволило сформировать надежную основу для дальнейшей проектно-конструкторской деятельности.

Моделирование и обоснование архитектуры системы

В рамках научно-исследовательского этапа была разработана и обоснована модель клиент-серверной архитектуры будущей системы управления тикетами. Эта архитектура была выбрана за ее способность к распределению нагрузки, упрощению масштабирования и повышению безопасности за счет изоляции серверной логики от клиентской. Основными компонентами модели стали:

- База данных (PostgreSQL): PostgreSQL был выбран в качестве основной СУБД благодаря своей надежности, обширному функционалу, поддержке ACID-транзакций и высокой производительности. В ходе исследований было обосновано использование следующих возможностей:

    - UUID-ключи: Для обеспечения глобальной уникальности идентификаторов тикетов, пользователей и других сущностей, что критически важно в распределенных или потенциально масштабируемых системах.

    - Типы CITEXT и JSONB: CITEXT для регистронезависимых текстовых полей (например, email), упрощающий операции сравнения, и JSONB для гибкого хранения неструктурированных или полуструктурированных данных (например, метаданных вложений или пользовательских настроек), что обеспечивает гибкость схемы.

    - Партиционирование по диапазону (PARTITION BY RANGE) по полю created_at: Это решение было исследовано для повышения производительности запросов и упрощения управления большими объемами данных. Разделение данных по временным диапазонам позволяет эффективно архивировать старые данные, ускорять запросы к свежим данным и оптимизировать операции обслуживания базы данных.

- Триггеры: Было обосновано применение триггеров для автоматизации рутинных операций и обеспечения целостности данных, включая:
    - Автоматическое обновление поля updated_at при изменении записи.
    - Обновление полнотекстового индекса (search_vector) при изменении контента, предназначенного для поиска, что критически важно для актуальности поисковой выдачи.
    - Ведение DDL-аудита, фиксирующего все изменения в схеме базы данных, что повышает безопасность и контролируемость.

- Бэкенд-сервер (Go): Язык Go был выбран для реализации бэкенда благодаря его эффективности, высокой производительности, мощной поддержке конкурентности и встроенным инструментам для создания масштабируемых сетевых приложений. Использование фреймворков Gin или Echo обеспечивает быструю разработку RESTful API, а библиотеки sqlx или GORM упрощают взаимодействие с PostgreSQL.

- Фронтенд (C++ с Qt6 Widgets): C++ и Qt6 были выбраны для разработки клиентской части приложения, обеспечивая высокую производительность, кроссплатформенность и возможность создания богатого пользовательского интерфейса. Применение QNetworkAccessManager для HTTP-запросов обеспечивает надежное взаимодействие с бэкендом, а QSortFilterProxyModel позволяет эффективно управлять и фильтровать данные в табличных представлениях без постоянных запросов к серверу.

Разработка алгоритмов и подходов к автоматизации

На основе выбранной архитектуры и технологического стека был сформирован детальный алгоритм решения задачи автоматизации процессов управления тикетами, включающий следующие ключевые аспекты:

- Алгоритм JWT-аутентификации: Проведено исследование механизмов токен-основанной аутентификации. Выбор JWT обусловлен его безгосударственностью, что упрощает масштабирование бэкенда, и возможностью безопасной передачи информации о пользователе и его ролях. Алгоритм включает этапы генерации токена при входе, его валидации при каждом запросе к защищенным ресурсам и механизм обновления/отзыва токенов.

- Проектирование CRUD-операций и бизнес-логики: Разработан комплекс алгоритмов для выполнения Create, Read, Update, Delete операций над основными сущностями системы (тикеты, пользователи, комментарии, вложения). Особое внимание уделено алгоритмам, обеспечивающим:
    - Управление ролями и правами доступа: Динамическое определение доступных действий для пользователей в зависимости от их роли.
    - Soft-delete: Алгоритм логического удаления записей путем пометки их специальным флагом (deleted_at), что позволяет сохранить исторические данные и упрощает восстановление.
    - Хранение истории изменений: Разработан механизм для фиксации всех значимых изменений в тикетах, обеспечивающий полную прозрачность жизненного цикла заявки.

- Алгоритм полнотекстового поиска: Исследованы различные подходы к реализации полнотекстового поиска. Принято решение использовать встроенные возможности PostgreSQL с расширением pg_trgm, позволяющим выполнять эффективный поиск по частичным совпадениям и триграммам, что значительно улучшает релевантность и скорость поисковых запросов по описаниям тикетов.

- Система аудита: Разработан детальный алгоритм ведения аудита DDL-операций в базе данных, который фиксирует все изменения структуры схемы, пользователя, время и детали операции. Это критически важно для обеспечения безопасности, отслеживания изменений и восстановления в случае непредвиденных ситуаций.

Результаты научно-исследовательского этапа сформировали надежную теоретическую и проектную базу, определив ключевые технические решения и подходы, которые были впоследствии реализованы на проектно-конструкторском этапе.

1.3. Проектно-конструкторский этап

На данном этапе практики была осуществлена детальная реализация спроектированной серверной части системы управления тикетами, которая базировалась на результатах всестороннего научно-исследовательского этапа. В ходе работы были разработаны ключевые программные компоненты бэкенд-сервера и базы данных, тщательно проведены процедуры тестирования и верификации их работоспособности в условиях, максимально приближенных к эксплуатационным.

Разработка программного обеспечения

Подготовка окружения и инфраструктуры бэкенда

Начальным и одним из фундаментальных этапов стала тщательная настройка среды разработки и развертывания бэкенд-инфраструктуры, строго соответствующая требованиям технического задания. Были инсталлированы и параметризованы:

- Go (версия 1.20+): Современная среда для разработки высокопроизводительного и масштабируемого бэкенд-сервера.

- PostgreSQL (версия 13+): Реляционная СУБД, выбранная за её надежность, соответствие стандартам SQL, обширный набор функций для работы с данными, включая поддержку сложных структур и оптимизированных запросов.

- golang-migrate: Инструмент для версионного управления схемой базы данных, позволяющий безопасно и контролируемо применять изменения к структуре БД, а также обеспечивающий возможность полного воспроизведения схемы "с нуля" при развертывании.

Вся архитектура бэкенда, включающая сервер API и СУБД PostgreSQL, была контейнеризирована с использованием Docker. Управление контейнерами осуществлялось посредством docker-compose, что значительно упростило процесс локального запуска, изоляции компонентов и обеспечило высокую степень переносимости решения между различными средами. Конфигурационные параметры, такие как строка подключения к базе данных (DB_URL) и секретный ключ для работы с JWT (JWT_SECRET), были вынесены в файл .env, что повышает гибкость настройки и безопасность хранения чувствительных данных, предваряя переход на более продвинутые системы управления секретами (Vault/KMS). Было реализовано использование TLS для всех HTTP-запросов, обеспечивая шифрование трафика. Пароли пользователей хранились в базе данных исключительно в виде надежных bcrypt-хешей, предотвращая их компрометацию.

Реализация бэкенд-сервера и бизнес-логики

Бэкенд-сервер был разработан на языке Go, используя фреймворк Gin для построения REST API. Выбор фреймворка был обусловлен способностью обеспечивать исключительную производительность, встроенные механизмы для создания надежных сетевых приложений. Архитектура сервера была спроектирована с учетом принципов модульности и инверсии зависимостей, что упростило тестирование и поддержку.

- Модели данных и миграции: Были тщательно спроектированы структуры данных (Go-модели), точно отражающие сущности системы в базе данных: users (пользователи), tickets (тикеты), comments (комментарии), attachments (вложения), а также вспомогательные сущности, такие как roles, permissions, ticket_history и audit_log. Для каждой модели были разработаны соответствующие миграции golang-migrate, охватывающие создание таблиц, определение связей, добавление индексов и настройку специализированных функций БД (триггеры, партиции).

Основой для хранения данных послужила реляционная модель, спроектированная с учётом принципов нормализации для минимизации избыточности и обеспечения целостности данных. Для визуализации структуры была разработана ER-диаграмма (Entity-Relationship Diagram), которая наглядно отображает ключевые сущности системы и связи между ними (Рисунок 1).

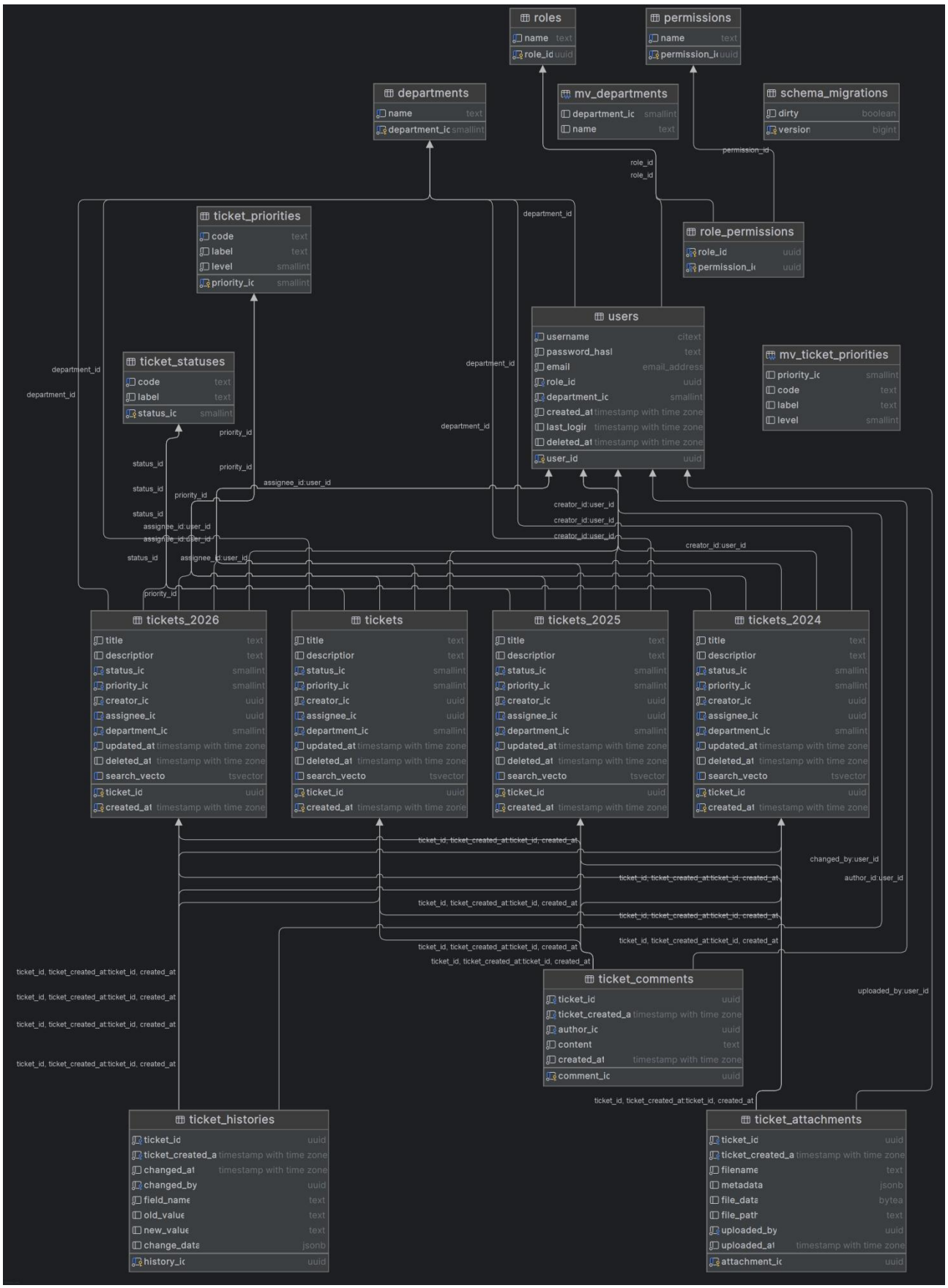

Рисунок 1 – ER-диаграмма базы данных системы управления тикетами

Ключевые сущности и связи на диаграмме:

1. users: Центральная таблица, хранящая данные о пользователях. Она связана с ролями (roles) и отделами (departments), что определяет права доступа и принадлежность пользователя.
2. tickets (партиционированная таблица): Основная сущность системы. Как видно на схеме, эта таблица партиционирована по годам (tickets_2024, tickets_2025 и т.д.). Такой подход значительно ускоряет запросы к актуальным данным и упрощает архивирование старых записей. Таблица связана с users через внешние ключи reporter_id (создатель) и assignee_id (исполнитель).
3. ticket_comments и ticket_attachments: Таблицы комментариев и вложений. Обе связаны с tickets отношением "многие к одному", позволяя каждому тикету иметь множество связанных записей.
4. ticket_histories: Таблица для аудита и логирования всех изменений в тикетах, обеспечивая полную прослеживаемость жизненного цикла заявки. Связана с таблицей tickets.
5. Справочные таблицы: ticket_statuses, ticket_priorities, departments служат как словари для стандартизации данных в тикетах.
6. Система прав доступа (RBAC): Реализована через таблицы roles, permissions и связующую таблицу role_permissions. Эта структура позволяет гибко настраивать права доступа для различных групп пользователей.
7. Материализованные представления (mv_...): Кэширование результатов запросов к справочникам, что существенно ускоряет получение часто запрашиваемых данных.

- CRUD-эндпоинты и API-дизайн: Для всех ключевых сущностей системы были реализованы стандартные CRUD-операции (Create, Read, Update, Delete) в строгом соответствии с принципами RESTful API. Каждый эндпоинт был разработан с учетом:
    1. Принципов REST: Использование стандартных HTTP-методов (GET, POST, PUT, DELETE) и ресурсов для выполнения операций.
    2. Обработки входящих данных: Валидация запросов и приведение типов.
    3. Взаимодействия с базой данных: Эффективное выполнение запросов к PostgreSQL.
    4. Формат ответа: Возврат данных в унифицированном формате JSON, а также соблюдение единого формата для обработки и возврата ошибок, что упрощает взаимодействие с клиентами API.

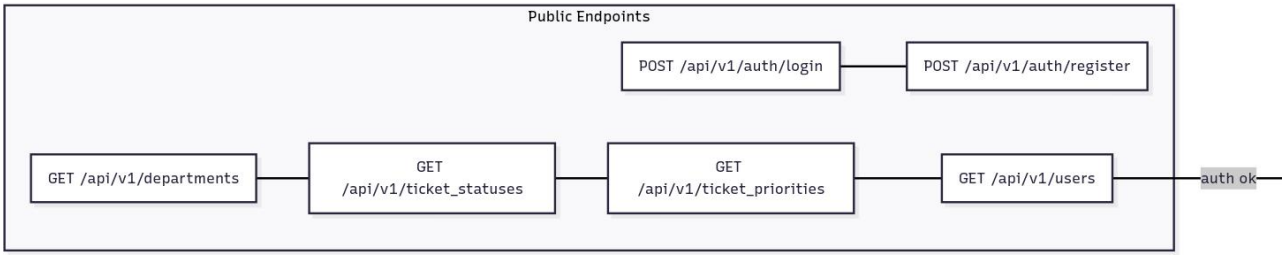

Рисунок 2 – Публичные CRUD-эндпоинты для сущностей departments, ticket_statuses, ticket_priorities, users

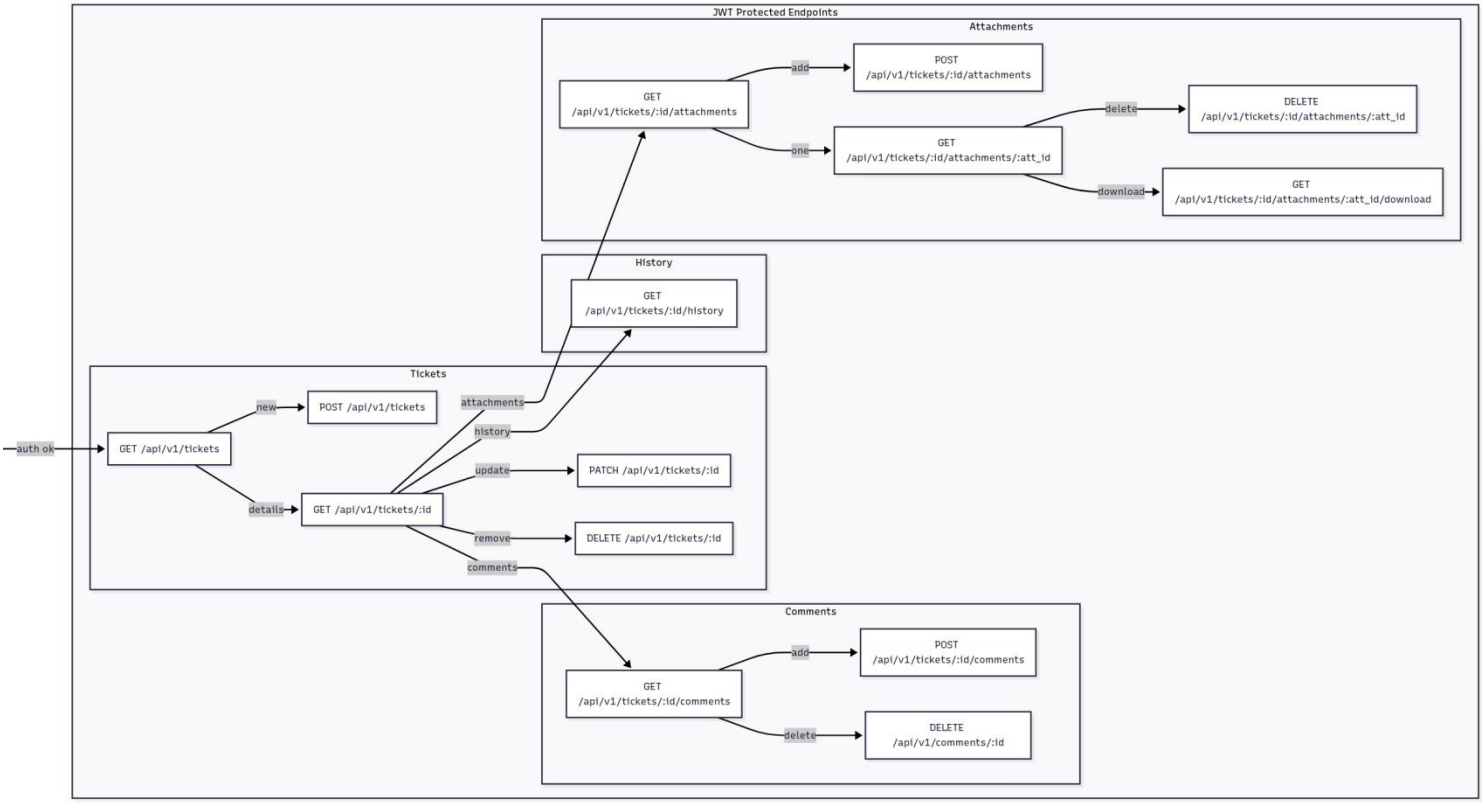

Рисунок 3 – Защищённые JWT (JSON Web Token) CRUD-эндпоинты для сущностей tickets, comments, history, attachments

- JWT-аутентификация и авторизация: Внедрена надежная система JWT-аутентификации для обеспечения безопасного доступа к ресурсам API. Это включало реализацию эндпоинтов для регистрации новых пользователей (/auth/register) и их входа в систему (/auth/login), а также механизм валидации JWT-токенов для защиты всех API-маршрутов, требующих авторизации. Система также обеспечивала детальное разграничение доступа к ресурсам в зависимости от ролей и прав, назначенных пользователям, что предотвращает несанкционированный доступ к конфиденциальным операциям.

- Бизнес-логика и расширенные возможности БД:
    - Soft-delete: Для сущностей, подлежащих логическому удалению (например, тикеты, комментарии), был реализован механизм "мягкого" удаления. Вместо физического удаления записи помечаются специальным флагом (deleted_at), что позволяет сохранить исторические данные и упрощает их восстановление при необходимости.

    - История изменений тикетов: Разработан автоматический механизм протоколирования всех значимых изменений в тикетах (например, изменение статуса, приоритета, ответственного) в отдельной таблице ticket_history. Это обеспечивает полную прослеживаемость и аудит жизненного цикла каждой заявки.

    - Полнотекстовый поиск: Для реализации высокоэффективного поиска по содержимому тикетов (по заголовку, описанию, комментариям) был задействован функционал PostgreSQL с расширением pg_trgm. Данное расширение позволило выполнять быстрый и релевантный поиск по тексту, включая частичные совпадения и опечатки, за счет использования триграммных индексов.

    - Триггеры и индексирование: Настроены сложные триггеры на уровне базы данных для автоматического выполнения рутинных, но критически важных операций:

      Автоматическое обновление поля updated_at (метки времени последнего изменения) при каждом изменении записи.

      Поддержание актуальности поля search_vector (специализированный тип данных для полнотекстового поиска) при изменении контента, что обеспечивает своевременное обновление поисковых индексов.

- Были созданы необходимые индексы (B-tree для первичных/внешних ключей, GIN для search_vector) для оптимизации выполнения запросов к базе данных, значительно ускоряя операции чтения.

- Партиционирование: Для оптимизации работы с большими объемами данных и повышения производительности запросов было реализовано партиционирование таблиц (в частности, tickets и ticket_history) по диапазону (PARTITION BY RANGE) на основе поля created_at. Это позволило более эффективно управлять данными, ускорять запросы к актуальным данным и упрощать процедуры архивирования старых записей.

- Материализованные представления: Для оптимизации доступа к часто запрашиваемым справочным данным (например, списки статусов, приоритетов, департаментов) были созданы материализованные представления. Они кэшируют результаты сложных запросов, значительно ускоряя их выполнение.

- DDL-аудит: Разработана всеобъемлющая система аудита DDL-команд на уровне базы данных, которая автоматически регистрировала все изменения в схеме (создание, изменение, удаление таблиц, индексов, функций), фиксируя пользователя, время и детальную информацию об операции. Это критически важно для обеспечения безопасности, отслеживания изменений и восстановления в случае непредвиденных ситуаций.

После реализации всех основных функциональных модулей бэкенда были проведены всесторонние мероприятия по проверке его работоспособности, стабильности, надежности и соответствия требованиям технического задания.

- Тестирование: Были разработаны и успешно выполнены unit-тесты и integration-тесты для всех ключевых компонентов бэкенда. Это включало тестирование API-эндпоинтов, логики взаимодействия с базой данных (CRUD-операции, работа с триггерами и партициями), модулей аутентификации, обработки бизнес-правил (роли, права, soft-delete, история), а также работы с файлами. Целевой показатель покрытия кода тестами ≥80% был достигнут, что подтверждает высокий уровень надежности и корректности реализованного функционала. В процессе разработки и тестирования активно использовался статический анализатор кода golangci-lint для выявления потенциальных ошибок и улучшения качества кода.

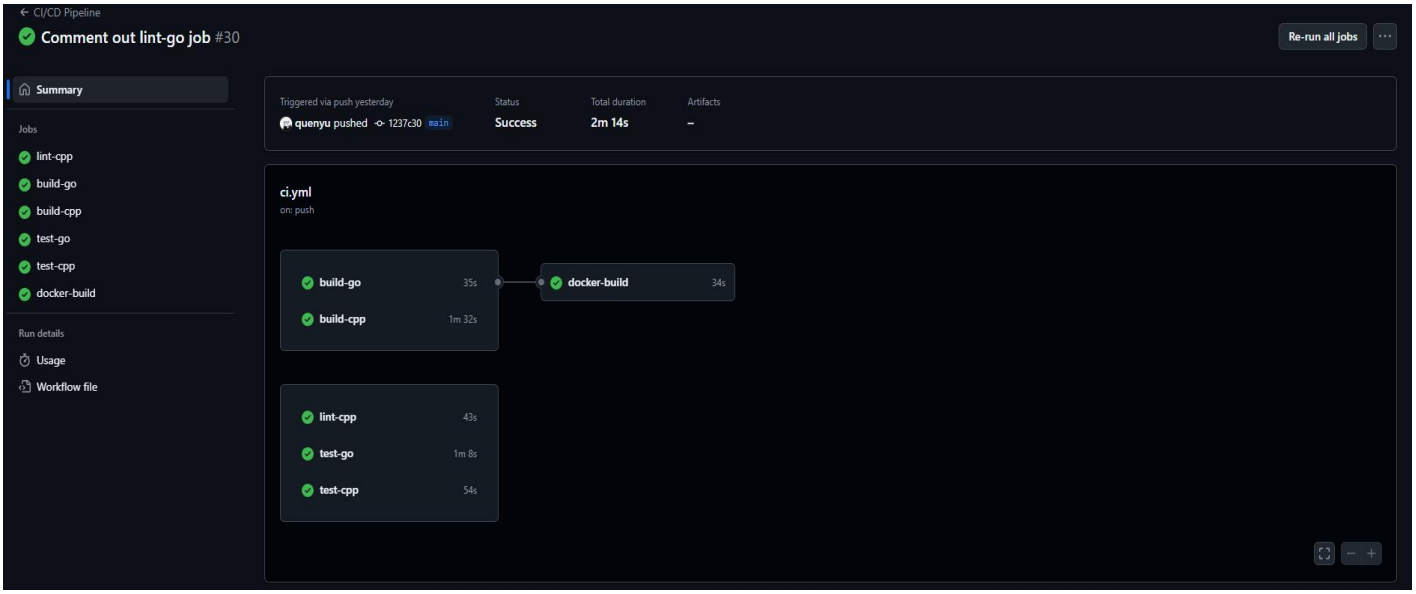

Рисунок 4 – Полное рабочие покрытие тестами (test-go), проверенное через GitHub Actions

- Документирование API: Все разработанные эндпоинты REST API были подробно и стандартизировано задокументированы с использованием спецификации Swagger. Это обеспечило четкое и машиночитаемое описание всех интерфейсов взаимодействия с бэкендом, включая методы, параметры запросов, форматы ответов и возможные ошибки. Наличие такой документации значительно упрощает процесс интеграции с любыми клиентскими приложениями и способствует дальнейшему развитию API.

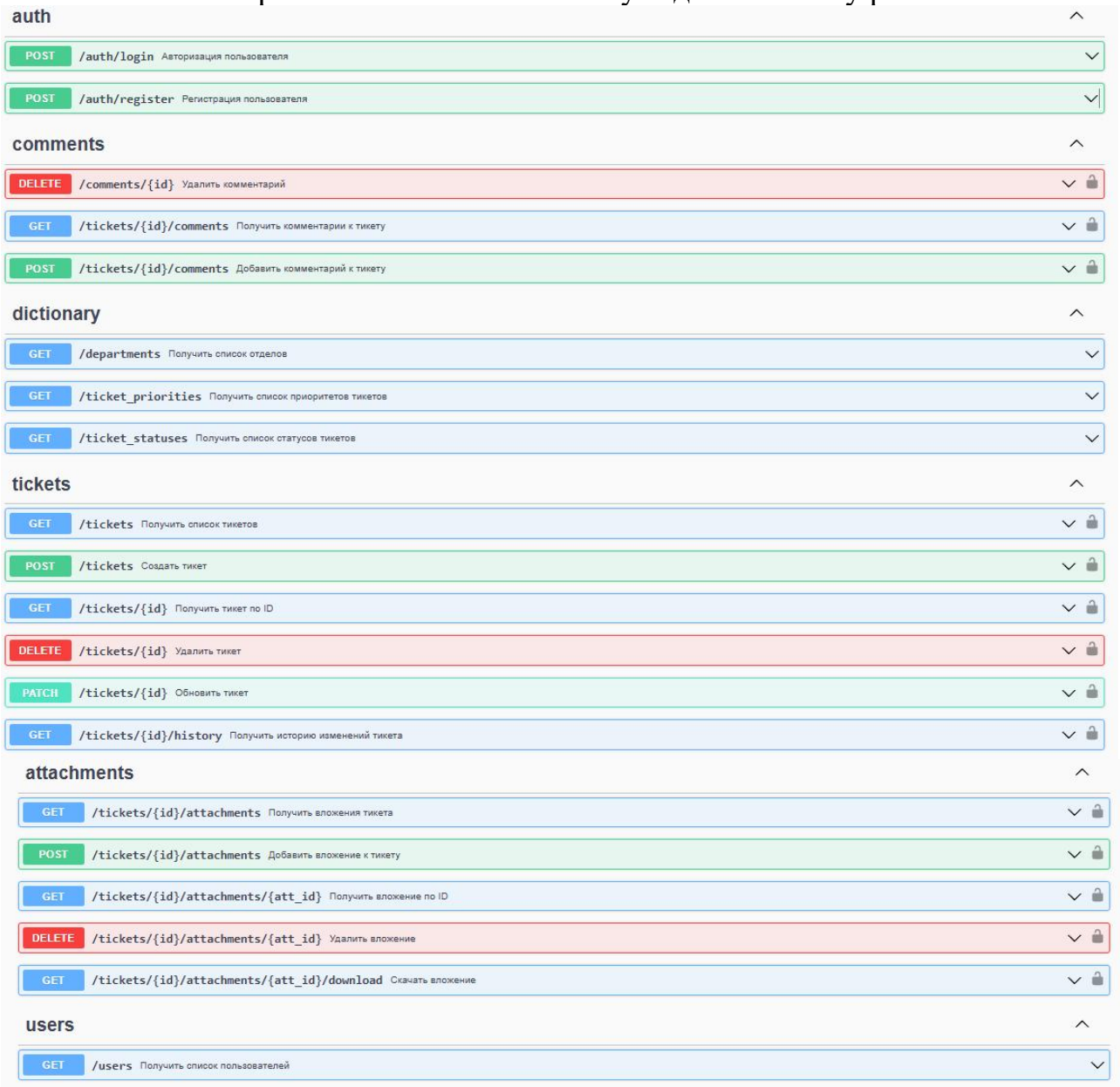

Рисунок 5 – Полностью задукоментированное RESTful API через swagger

- Верификация развертывания и CI/CD: Особое внимание было уделено проверке возможности развертывания всей бэкенд-системы в новом окружении. Скрипты миграций golang-migrate были успешно протестированы, подтвердив корректное и безошибочное создание всех таблиц, триггеров, партиций и структуры аудита в чистой базе данных. Полный стек бэкенда (сервер API и PostgreSQL) успешно запускался и функционировал через docker-compose, что подтвердило корректность настройки контейнеризации и готовность решения к быстрому и надежному развертыванию на любой совместимой серверной инфраструктуре. Были также настроены базовые пайплайны CI/CD (GitHub Actions), включающие автоматический запуск линтеров, юнит-тестов, сборку Docker-образа бэкенда и его публикацию в приватный registry.

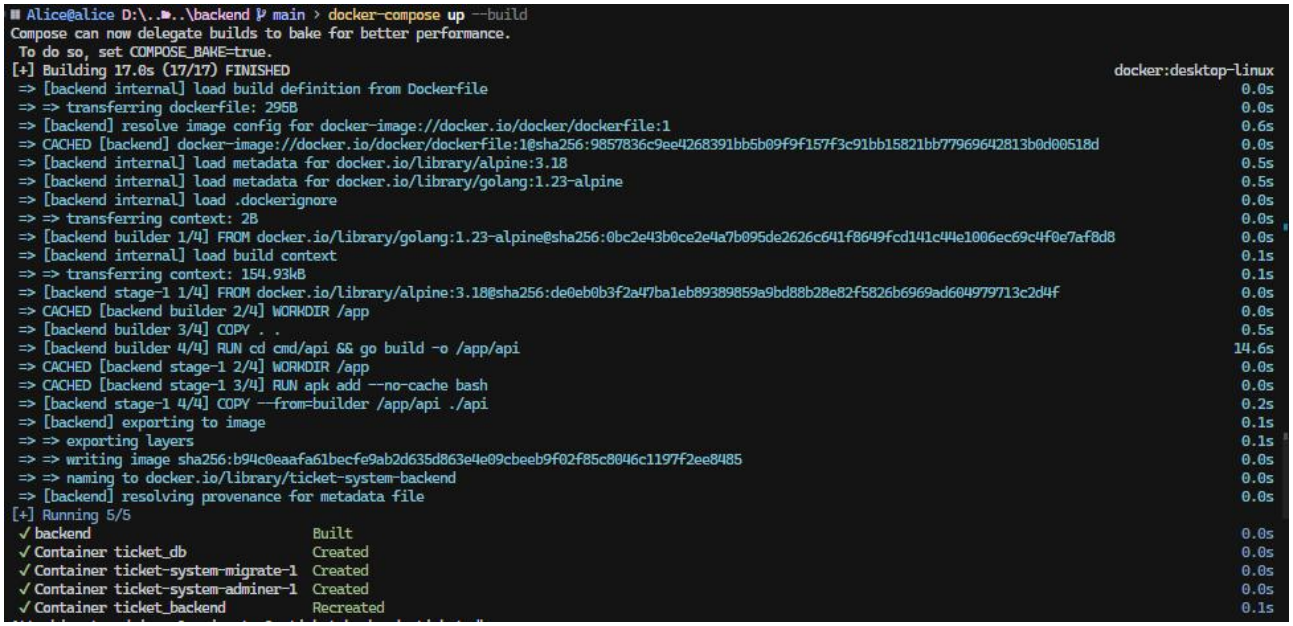

Рисунок 6 – Рабочая сборка Docker-образа на основе docker-compose.yml файла

Анализ результатов и обобщение

На проектно-конструкторском этапе была успешно реализована полноценная серверная часть системы управления тикетами, полностью соответствующая требованиям индивидуального задания и технического задания, с акцентом на бэкенд-функционале и взаимодействии с базой данных. Выбранные технологии (Go, PostgreSQL, Docker) и архитектурные решения доказали свою высокую эффективность, обеспечив необходимую функциональность, производительность, безопасность и надежность. Разработанное программное обеспечение бэкенда демонстрирует зрелость и готовность к дальнейшему развитию, масштабированию и интеграции с различными клиентскими приложениями.

ЗАКЛЮЧЕНИЕ

В ходе выполнения учебной, ознакомительной практики была успешно решена задача по разработке серверной части клиент-серверного модуля управления тикетами. Данный проект позволил достичь основной цели практики — сформировать, закрепить и развить практические навыки в области системного анализа, проектирования и разработки сложного программного обеспечения.

В процессе работы были решены следующие ключевые задачи:

1. Получены структурированные сведения о предприятии: Проведён сбор и анализ информации о деятельности предприятия – базы практики, его организационной структуре и технологических процессах, что позволило погрузиться в контекст разработки прикладного программного обеспечения в условиях реального бизнеса.
2. Подготовлено полноценное рабочее окружение: Настроены все необходимые инструменты и технологии, включая Go, PostgreSQL, Docker и систему миграций базы данных. Создан файл конфигурации .env, обеспечивающий централизованное управление параметрами среды.
3. Реализован прототип серверной части: Разработан базовый REST API с использованием фреймворка Gin, интегрирована JWT-аутентификация. Созданы модели и миграции для ключевых сущностей (users, tickets, comments, attachments), реализованы основные CRUD-операции.
4. Внедрена бизнес-логика и системы безопасности: Настроены роли и права доступа, реализованы soft-delete, аудит изменений, полнотекстовый поиск, а также механизмы обновления временных меток и DDL-триггеры. Оптимизация производительности обеспечена с помощью индексирования и партиционирования таблиц.
5. Проведено тестирование и документирование: Написаны unit- и integration-тесты, обеспечившие покрытие кода свыше 80%. Описан и опубликован API с использованием Swagger. Подтверждена стабильная работа миграций и сборки проекта через docker-compose.
6. Проанализирован итоговый результат: По завершении работы подготовлен отчёт, в котором отражены этапы реализации, выявлены сложности, описаны принятые проектные решения и обоснована эффективность выбранного подхода.

Итогом практики является полностью работоспособный, протестированный и задокументированный бэкенд-сервис, готовый к интеграции с клиентским приложением и дальнейшему развитию. Все пункты индивидуального задания были выполнены в полном объёме.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. 1. Деменков, Н. П. Управление в технических системах : учебное пособие / Н. П. Деменков, Е. А. Микрин. — Москва : МГТУ им. Баумана, 2017. — 452 с. — ISBN 978-5-7038-4661-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/106397

2. Булдакова Т. И., Миков Д. А. Теория систем и системный анализ : учебно-методическое пособие / Булдакова Т. И., Миков Д. А. ; МГТУ им. Н. Э. Баумана (национальный исследовательский ун-т). - М. : Изд-во МГТУ им. Н. Э. Баумана, 2021. - 45 с. : рис., табл. - Библиогр.: с. 39. - ISBN 978-5-7038-5728-1.

3. Булдакова Т. И. Исследование сложных систем и процессов : учеб. пособие / Булдакова Т. И. ; МГТУ им. Н. Э. Баумана. - М. : Изд-во МГТУ им. Н. Э. Баумана, 2017. - 162 с. : ил. - Библиогр.: с. 158-161. - ISBN 978-5-7038-4511-0.

4. Кузнецов, В. В. Системный анализ : учебник и практикум для вузов / В. В. Кузнецов, А. Ю. Шатраков ; под общей редакцией В. В. Кузнецова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 333 с. — (Высшее образование). — ISBN 978-5-534-16199-1.

5. Волкова, В. Н. Теория систем и системный анализ : учебник для вузов / В. Н. Волкова, А. А. Денисов. — 3-е изд. — Москва : Издательство Юрайт, 2024. — 562 с. — (Высшее образование). — ISBN 978-5-534-14945-6.

6. Алексеева, М. Б. Теория систем и системный анализ : учебник и практикум для вузов / М. Б. Алексеева, П. П. Ветренко. — Москва : Издательство Юрайт, 2024. — 298 с. — (Высшее образование). — ISBN 978-5-534-17987-3.

7. Востриков, А. С. Теория автоматического регулирования : учебник и практикум для вузов / А. С. Востриков, Г. А. Французова. — Москва : Издательство Юрайт, 2024. — 279 с. — (Высшее образование). — ISBN 978-5-534-04845-2.

8. Волкова, В. Н. Управление в открытых системах : учебник для вузов / В. Н. Волкова. — Москва : Издательство Юрайт, 2024. — 573 с. — (Высшее образование). — ISBN 978-5-534-18060-2.

9. Ким, Д. П. Теория автоматического управления. Линейные системы : учебник и практикум для вузов / Д. П. Ким. — 3-е изд., испр. и доп. — Москва : Издательство Юрайт, 2024. — 311 с. — (Высшее образование). — ISBN 978-5-534-00799-2.

10. Ким, Д. П. Теория автоматического управления : учебник и практикум для вузов / Д. П. Ким. — Москва : Издательство Юрайт, 2024. — 276 с. — (Высшее образование). — ISBN 978-5-9916-9294-6.

11. Акопов, А. С. Имитационное моделирование : учебник и практикум для вузов / А. С. Акопов. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2024. — 426 с. — (Высшее образование). — ISBN 978-5-534-18379-5.

12. Боев, В. Д. Имитационное моделирование систем : учебное пособие для вузов / В. Д. Боев. — Москва : Издательство Юрайт, 2024. — 253 с. — (Высшее образование). — ISBN 978-5-534-04734-9.

13. Зализняк, В. Е. Введение в математическое моделирование : учебное пособие для вузов / В. Е. Зализняк, О. А. Золотов. — Москва : Издательство Юрайт, 2024. — 133 с. — (Высшее образование). — ISBN 978-5-534-12249-7.

14. Зализняк, В. Е. Введение в математическое моделирование : учебное пособие для вузов / В. Е. Зализняк, О. А. Золотов. — Москва : Издательство Юрайт, 2024. — 133 с. — (Высшее образование). — ISBN 978-5-534-12249-7.

15. Древс, Ю. Г. Имитационное моделирование : учебное пособие для вузов / Ю. Г. Древс, В. В. Золотарёв. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2024. — 142 с. — (Высшее образование). — ISBN 978-5-534-11385-3.

Дополнительные материалы

16. ГОСТ Р 57700.37–2021 «Компьютерные модели и моделирование. ЦИФРОВЫЕ ДВОЙНИКИ ИЗДЕЛИЙ. Общие положения».

17. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.

18. ГОСТ Р 57188 -2016 «ЧИСЛЕННОЕ МОДЕЛИРОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ. Термины и определения».