СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью производственной практики являлось закрепление теоретических знаний и получение практических навыков администрирования информационных ресурсов в условиях реально действующего ИТ-предприятия. Основной фокус был направлен на изучение архитектуры корпоративных информационных систем, принципов обеспечения их бесперебойной работы и безопасности.

Для достижения поставленной цели в период практики решались следующие задачи: анализ ИТ-инфраструктуры организации и ее ключевых компонентов, изучение нормативной базы, регламентирующей обработку данных, а также непосредственное участие в повседневных задачах по поддержке и оптимизации работы информационных ресурсов компании.

ООО "Малленом Системс" – ведущий российский разработчик программно-аппаратных решений в области промышленного машинного зрения и интеллектуальной видеоаналитики. Компания специализируется на создании высокотехнологичных систем, использующих алгоритмы компьютерного зрения (CV), машинного обучения (ML) и глубокого обучения (Deep Learning) для автоматизации критически важных задач контроля качества, безопасности, логистики и учета на промышленных предприятиях, транспортных узлах и объектах инфраструктуры. Портфель продуктов компании охватывает решения для идентификации объектов (продукция, вагоны, автотранспорт), контроля технологических процессов, ситуационной безопасности и анализа транспортных потоков.

Сроки практики с 01.12.2025 по 14.01.2025, место прохождения практики ООО “Малленом Системс” - Металлургов 21б.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ОРГАНИЗАЦИИ

1.1 Краткая характеристика организации

ООО «Малленом Системс» является российским разработчиком программно-аппаратных комплексов в области промышленного машинного зрения и видеоаналитики. Компания занимает нишу высокотехнологичных решений для автоматизации контроля качества, безопасности и учета на крупных промышленных объектах. Ключевыми заказчиками выступают предприятия металлургической, машиностроительной и транспортной отраслей.

Организационная структура компании построена по матричному принципу, что характерно для проектно-ориентированных ИТ-фирм. Можно выделить несколько ключевых блоков: блок разработки (включающий frontend, backend и ML-инженеров), блок внедрения и технической поддержки, а также административно-хозяйственный блок. Непосредственное взаимодействие в период практики осуществлялось с отделом инфраструктуры и системными администраторами, обеспечивающими работу как внутренних сервисов, так и продуктовых сред заказчиков.

1.2 Роль информационных ресурсов и ИТ-инфраструктуры в работе организации

В компании «Малленом Системс» ИТ-инфраструктура является не просто вспомогательным инструментом, а фундаментом основного бизнеса. Вся деятельность - от обучения нейросетей до демонстрации готовых решений заказчику - строится на базе вычислительных мощностей и сетевого взаимодействия.

ИТ-инфраструктура организации представляет собой гибридную среду. Основная вычислительная нагрузка ложится на собственный on-premise кластер GPU-серверов (на базе NVIDIA A100 и A40), объединенных высокоскоростной сетью InfiniBand. Эти мощности критически важны для параллельного обучения моделей глубокого обучения. Для хранения данных используется многоуровневая система: высокопроизводительные NVMe-массивы для активных датасетов и объектное хранилище (на базе MinIO) для архивирования сырых видеоданных. Корпоративная сеть построена на оборудовании Cisco и разделена на виртуальные сегменты (VLAN) для изоляции трафика разработки, административного учета и гостевого доступа.

Что касается информационных ресурсов, то в компании активно используются как тиражируемые, так и внутренние разработки. Для управления взаимоотношениями с клиентами и учета лидов применяется CRM-система (Bitrix24), интегрированная с корпоративной телефонией. Управление проектами и технической документацией централизовано в Atlassian Confluence и Jira. Однако ядром информационных ресурсов являются специализированные СУБД: реляционная PostgreSQL используется для транзакционных систем, а векторная база данных (Milvus) задействована для хранения эмбеддингов изображений, что позволяет быстро осуществлять семантический поиск по визуальному контенту.

1.3 Цели и задачи перед вами на период практики

Индивидуальное задание на период практики было направлено на изучение текущего состояния ИТ-инфраструктуры и участие в рутинных операциях по ее поддержке. Основной целью для меня ставилось ознакомление с полным циклом администрирования информационных ресурсов - от мониторинга их состояния до обеспечения отказоустойчивости.

В соответствии с этим были определены следующие задачи: во-первых, провести инвентаризацию и анализ документооборота, касающегося регламентов работы с данными. Во-вторых, освоить инструменты мониторинга и резервного копирования, используемые в компании. В-третьих, принять участие в текущей деятельности отдела - от обработки заявок пользователей до помощи в настройке тестовых сред для разработчиков.

1.4 Нормативные документы

Работа с информационными ресурсами в ООО «Малленом Системс» строго регламентируется как требованиями российского законодательства, так и внутренними локальными актами. Базовыми документами федерального уровня выступают Федеральный закон № 149-ФЗ «Об информации...» и № 152-ФЗ «О персональных данных», что особенно актуально, поскольку компания работает с биометрическими данными в составе видеопотоков. Наличие статуса аккредитованной ИТ-компании также накладывает обязательства по ведению отчетности в соответствии с требованиями Минцифры.

Внутренняя документация представлена целым рядом регламентов. Ключевым документом является Политика информационной безопасности, которая устанавливает правила управления доступом и классификации информации (открытая, конфиденциальная, строго конфиденциальная). Для сотрудников отдела инфраструктуры обязательными к исполнению являются регламенты резервного копирования (политика 3-2-1) и регламент управления инцидентами, описывающий порядок действий при сбоях в работе сервисов.

2 АДМИНИСТРИРОВАНИЕ ИНФОРМАЦИОННЫХ РЕСУРСОВ

2.1 Информационные ресурсы и инфраструктура

Администрирование столь разнородной инфраструктуры требует применения специализированных инструментов. Управление серверами в «Малленом Системс» строится на сочетании ручного контроля и автоматизации. Для управления конфигурациями активно используется система Ansible, с помощью которой автоматизирована первоначальная настройка новых серверов и обновление пакетов на парке Linux-машин.

Вопрос резервного копирования решен через использование Veeam Backup & Replication. Мне довелось ознакомиться с политикой резервирования, которая строится по правилу «3-2-1»: три копии данных хранятся на двух разных носителях (быстрый диск и ленточная библиотека), причем одна копия выгружается в облачное S3-хранилище для защиты от физического повреждения серверной.

Мониторинг работоспособности систем построен на связке Prometheus и Grafana. Эти инструменты позволяют в реальном времени отслеживать критические метрики: загрузку GPU-ядер при обучении моделей, заполнение дисковых массивов и сетевую задержку. В случае выхода параметров за допустимые пределы в мессенджер дежурной смены приходит автоматическое оповещение, что позволяет реагировать на проблемы до того, как они затронут пользователей.

2.2 Безопасность информационных ресурсов

Информационная безопасность в компании обеспечивается комплексным подходом, сочетающим защиту периметра и контроль доступа к внутренним ресурсам. На границе сети установлен межсетевой экран

следующего поколения (NGFW) UserGate, который фильтрует трафик и обеспечивает работу VPN для удаленных сотрудников. Для конечных точек используется Kaspersky Endpoint Security с централизованным управлением.

Организация доступа построена на ролевой модели (RBAC) с использованием Active Directory. Каждый сотрудник получает ровно тот объем прав, который необходим для выполнения его непосредственных обязанностей. Критически важным является требование обязательной двухфакторной аутентификации (MFA) при доступе к production-серверам и консолям управления облачными сервисами. За время практики серьезных инцидентов безопасности не произошло, однако на еженедельных летучках разбирались учебные примеры фишинговых атак, что подчеркивает важность обучения персонала. Компания регулярно проводит внутренние тренинги по кибергигиене, а для ИТ-отдела организуются более глубокие семинары по безопасной разработке.

2.3 Автоматизация и оптимизация процессов

В рамках прохождения практики я наблюдал и участвовал в процессах автоматизации рутинных задач. В компании уже автоматизированы такие процессы как деплоймент (развертывание) типовых сервисов с помощью GitLab CI/CD и сбор логов с использованием стека Elasticsearch (ELK). Это позволяет разработчикам не тратить время на настройку окружения, а администраторам - быстро находить причины ошибок.

Мое личное участие выразилось в помощи по оптимизации процесса резервного копирования конфигураций сетевых устройств. Было замечено, что бэкапы коммутаторов делаются нерегулярно. Мной был предложен и реализован (под руководством наставника) скрипт на Python, который подключается по SSH к оборудованию, снимает конфигурацию и складывает

ее в репозиторий Git. Это позволило автоматизировать ручную работу и добавить контроль версий.

Кроме того, в ходе практики я участвовал в анализе проблемы медленной работы одного из тестовых веб-сервисов. Вместе с наставником мы изучили логи, выявили неоптимальные запросы к базе данных. Хотя исправление запросов - задача разработчиков, мною было предложено настроить кэширование ответов с использованием Redis на время тестирования, что временно, но ощутимо снизило нагрузку на сервер.

3 ВЫПОЛНЯЕМЫЕ ЗАДАНИЯ

В ходе прохождения производственной практики одной из ключевых задач стала разработка прототипа веб-приложения для мониторинга состояния информационной безопасности серверной инфраструктуры. Необходимость создания подобного инструмента была обусловлена потребностью в централизованном управлении сведениями о серверах, политиках безопасности и событиях ИБ, а также визуализации основных показателей в удобном интерфейсе.

Разработанное решение представляет собой веб-приложение, реализованное на языке Python с использованием фреймворка FastAPI. В качестве ORM использовалась библиотека SQLAlchemy, обеспечивающая взаимодействие с реляционной базой данных SQLite. Пользовательский интерфейс построен на базе шаблонизатора Jinja2 и CSS-фреймворка Bootstrap версии 5.3.3. Для визуализации статистических данных применялась библиотека Chart.js.

Исходный код основных маршрутов приложения представлен на Рисунке 1, структура проекта - на Рисунке 2.

Архитектура и функциональность решения

Приложение построено по классической трёхуровневой архитектуре:

- уровень представления (HTML-шаблоны),
- уровень бизнес-логики (маршруты FastAPI),
- уровень хранения данных (SQLite через SQLAlchemy).

На главной странице (Dashboard) реализована панель мониторинга с ключевыми показателями (KPI): общее количество серверов, активных политик безопасности и зарегистрированных событий. Пример интерфейса панели мониторинга представлен на Рисунке 3.

Функциональность приложения включает:

- добавление, редактирование и удаление серверов инфраструктуры;

– управление политиками безопасности;
– регистрацию и отслеживание событий информационной безопасности;
– фильтрацию событий по уровню критичности;
– визуализацию статистики.

Интерфейс управления серверами представлен на Рисунке 4. Раздел управления политиками безопасности приведён на Рисунке 5. Страница логирования событий безопасности представлена на Рисунке 6.

Реализованные модели данных

В рамках разработки были спроектированы и реализованы следующие сущности базы данных:

1. Server - содержит сведения о сервере (имя, IP-адрес, статус).
2. SecurityPolicy - описывает политику безопасности.
3. SecurityEvent - фиксирует событие ИБ с указанием сервера, политики и уровня критичности.
4. Service - отражает состояние отдельных сервисов на сервере.

Связи между сущностями реализованы средствами SQLAlchemy через внешние ключи, что позволило обеспечить целостность данных и корректное отображение взаимосвязей в интерфейсе.

Практические задачи, выполненные в ходе разработки

В процессе выполнения задания были решены следующие практические задачи:

– проектирование структуры базы данных;
– реализация CRUD-операций для всех сущностей;
– настройка маршрутизации в FastAPI;
– разработка HTML-шаблонов с использованием Jinja2;
– реализация визуализации статистики через Chart.js;
– тестирование корректности обработки пользовательских данных;
– заполнение базы тестовыми данными с помощью вспомогательного скрипта.

Особое внимание уделялось корректности обработки входных данных и предотвращению логических ошибок при создании связей между объектами (например, невозможность регистрации события без привязки к серверу).

Трудности и способы их решения

В ходе разработки возник ряд технических сложностей.

Одной из проблем стала корректная организация связей между моделями SQLAlchemy. На раннем этапе разработки возникали ошибки при удалении записей, связанных внешними ключами. Проблема была решена путём настройки каскадного поведения и корректной конфигурации отношений (relationship).

Дополнительные трудности были связаны с отображением статистики на Dashboard. Первоначально данные передавались в шаблон некорректно, что приводило к ошибкам JavaScript при построении графиков. После анализа структуры JSON-объектов и проверки формата передаваемых данных проблема была устранена.

Также требовала внимания логика фильтрации событий по уровню серьёзности. Была реализована дополнительная проверка параметров запроса и обработка возможных некорректных значений.

Все проблемы решались путём тестирования отдельных модулей, изучения официальной документации используемых библиотек и последовательной отладки кода.

Освоенные инструменты и технологии

В ходе практики были углублены знания и практические навыки работы со следующими технологиями:

- разработка REST-приложений на FastAPI;
- работа с ORM SQLAlchemy;
- проектирование реляционных баз данных;
- построение серверных HTML-шаблонов (Jinja2);
- использование Bootstrap для адаптивной верстки;

– интеграция JavaScript-библиотек (Chart.js);
– управление зависимостями Python-проекта;
– тестирование и локальный запуск приложения через Uvicorn.

Дополнительно были закреплены навыки работы в Linux-среде, использования Git для контроля версий и структурирования проекта по модульному принципу.

Взаимодействие с другими подразделениями

Разработка приложения осуществлялась во взаимодействии с сотрудниками отдела инфраструктуры и информационной безопасности.

С отделом ИБ согласовывалась структура карточек KPI и перечень параметров, которые целесообразно отображать на панели мониторинга. Это позволило приблизить прототип к реальным задачам управления безопасностью.

Системные администраторы консультировали по типовой структуре серверной инфраструктуры и формату представления информации о статусах серверов.

Таким образом, выполнение задания носило комплексный характер и позволило не только реализовать программное решение, но и отработать навыки проектирования, разработки и согласования технических решений в рамках промышленной ИТ-организации.

ЗАКЛЮЧЕНИЕ

В ходе прохождения производственной практики в ООО «Малленом Системс» была изучена структура и принципы функционирования ИТ-инфраструктуры компании, специализирующейся на разработке программно-аппаратных решений в области промышленного машинного зрения и интеллектуальной видеоаналитики. Практика позволила получить целостное представление о процессах администрирования информационных ресурсов в условиях высокотехнологичной ИТ-организации.

В рамках поставленных задач был проведён анализ архитектуры инфраструктуры, механизмов мониторинга, резервного копирования и разграничения доступа. Особое внимание уделялось вопросам информационной безопасности, поскольку специфика деятельности компании связана с обработкой чувствительных данных и обеспечением отказоустойчивости вычислительных мощностей.

Ключевым практическим результатом стала разработка прототипа веб-приложения для мониторинга состояния информационной безопасности серверной инфраструктуры. Реализованная система позволила централизовать сведения о серверах, политиках безопасности и событиях ИБ, а также визуализировать основные показатели в формате панели мониторинга. В процессе разработки были применены современные инструменты backend-разработки, ORM-технологии и средства визуализации данных.

Полученный опыт подтвердил важность интеграции автоматизации и инструментов разработки в процессы администрирования инфраструктуры. Практика показала, что эффективное управление ИТ-ресурсами требует сочетания технических навыков, понимания принципов информационной безопасности и умения взаимодействовать с различными подразделениями компании.

Теоретические знания, полученные в ходе обучения, в целом соответствуют базовым требованиям к специалисту в области администрирования информационных ресурсов. Однако практика продемонстрировала необходимость более глубокого изучения современных инструментов веб-разработки, автоматизации и обеспечения безопасности ИТ-систем.

В целом производственная практика способствовала формированию профессиональных компетенций в области администрирования информационных ресурсов, разработки прикладных инструментов мониторинга и понимания реальных процессов функционирования ИТ-инфраструктуры промышленной компании.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Об информации, информационных технологиях и о защите информации : федер. закон Рос. Федерации от 27.07.2006 № 149-ФЗ (ред. действующая на момент обращения) // Официальный интернет-портал правовой информации. – URL: http://pravo.gov.ru (дата обращения: 28.02.2026).

2. О персональных данных : федер. закон Рос. Федерации от 27.07.2006 № 152-ФЗ (ред. действующая на момент обращения) // Официальный интернет-портал правовой информации. – URL: http://pravo.gov.ru (дата обращения: 28.02.2026).

3. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection - Information security management systems - Requirements. – Geneva: International Organization for Standardization, 2022.

4. FastAPI : documentation. – URL: https://fastapi.tiangolo.com/ (дата обращения: 28.02.2026).

5. SQLAlchemy : documentation. – URL: https://docs.sqlalchemy.org/ (дата обращения: 28.02.2026).

6. SQLite : official documentation. – URL: https://www.sqlite.org/docs.html (дата обращения: 28.02.2026).

7. Jinja2 : documentation. – URL: https://jinja.palletsprojects.com/ (дата обращения: 28.02.2026).

8. Bootstrap 5.3 : documentation. – URL: https://getbootstrap.com/docs/5.3/ (дата обращения: 28.02.2026).

9. Chart.js : documentation. – URL: https://www.chartjs.org/docs/ (дата обращения: 28.02.2026).

10. Fielding R. T. Architectural styles and the design of network-based software architectures : doctoral dissertation. – Irvine: University of California, 2000.

```python
from fastapi import FastAPI, Request, Form, Depends
from fastapi.templating import Jinja2Templates
from fastapi.staticfiles import StaticFiles
from sqlalchemy.orm import Session
from database import init_db, SessionLocal
from models import Server, SecurityPolicy, SecurityEvent

app = FastAPI(title="Администрирование ИБ — MVP")
init_db()

templates = Jinja2Templates(directory="templates")

# зависимость для сессии
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

# dashboard
@app.get("/")
def dashboard(request: Request, db: Session = Depends(get_db)):
    servers_count = db.query(Server).count()
    policies_count = db.query(SecurityPolicy).count()
    events_count = db.query(SecurityEvent).count()
    return templates.TemplateResponse("dashboard.html", {
        "request": request,
        "servers_count": servers_count,
        "policies_count": policies_count,
        "events_count": events_count
    })

# servers
@app.get("/servers")
def servers_list(request: Request, db: Session = Depends(get_db)):
    servers = db.query(Server).all()
    return templates.TemplateResponse("servers.html", {"request": request, "servers": servers})

@app.get("/servers/create")
def server_create_form(request: Request):
    return templates.TemplateResponse("server_form.html", {"request": request, "action": "create"})

@app.post("/servers/create")
def server_create(request: Request, name: str = Form(...), ip_address: str = Form(...), status: str = Form("online")
    server = Server(name=name, ip_address=ip_address, status=status)
    db.add(server)
    db.commit()
    return templates.TemplateResponse("servers.html", {"request": request, "servers": db.query(Server).all()})

@app.get("/servers/edit/{server_id}")
def server_edit_form(request: Request, server_id: int, db: Session = Depends(get_db)):
    server = db.query(Server).filter(Server.id == server_id).first()
    return templates.TemplateResponse("server_form.html", {"request": request, "server": server, "action": "edit"})

@app.post("/servers/edit/{server_id}")
def server_edit(request: Request, server_id: int, name: str = Form(...), ip_address: str = Form(...), status: str =
    server = db.query(Server).filter(Server.id == server_id).first()
    server.name = name
    server.ip_address = ip_address
    server.status = status
    db.commit()
    return templates.TemplateResponse("servers.html", {"request": request, "servers": db.query(Server).all()})

@app.post("/servers/delete/{server_id}")
def server_delete(server_id: int, db: Session = Depends(get_db), request: Request = None):
```

Рисунок 1 - Исходный код основных маршрутов приложения часть 1.

```python
def server_delete(server_id: int, db: Session = Depends(get_db), request: Request = None):
    server = db.query(Server).filter(Server.id == server_id).first()
    db.delete(server)
    db.commit()
    servers = db.query(Server).all()
    return templates.TemplateResponse("servers.html", {"request": request, "servers": servers})

# policies
@app.get("/policies")
def policies_list(request: Request, db: Session = Depends(get_db)):
    policies = db.query(SecurityPolicy).all()
    return templates.TemplateResponse("policies.html", {"request": request, "policies": policies})

@app.get("/policies/create")
def policy_create_form(request: Request):
    return templates.TemplateResponse("policy_form.html", {"request": request, "action": "create"})

@app.post("/policies/create")
def policy_create(request: Request, name: str = Form(...), description: str = Form(""), db: Session = Depends(get_db
    policy = SecurityPolicy(name=name, description=description)
    db.add(policy)
    db.commit()
    return templates.TemplateResponse("policies.html", {"request": request, "policies": db.query(SecurityPolicy).all

@app.get("/policies/edit/{policy_id}")
def policy_edit_form(request: Request, policy_id: int, db: Session = Depends(get_db)):
    policy = db.query(SecurityPolicy).filter(SecurityPolicy.id==policy_id).first()
    return templates.TemplateResponse("policy_form.html", {"request": request, "policy": policy, "action": "edit"})

@app.post("/policies/edit/{policy_id}")
def policy_edit(request: Request, policy_id: int, name: str = Form(...), description: str = Form(""), db: Session =
    policy = db.query(SecurityPolicy).filter(SecurityPolicy.id==policy_id).first()
    policy.name = name
    policy.description = description
    db.commit()
    return templates.TemplateResponse("policies.html", {"request": request, "policies": db.query(SecurityPolicy).all

@app.post("/policies/delete/{policy_id}")
def policy_delete(request: Request, policy_id: int, db: Session = Depends(get_db)):
    policy = db.query(SecurityPolicy).filter(SecurityPolicy.id==policy_id).first()
    db.delete(policy)
    db.commit()
    return templates.TemplateResponse("policies.html", {"request": request, "policies": db.query(SecurityPolicy).all

# events
@app.get("/events")
def events_list(request: Request, db: Session = Depends(get_db)):
    events = db.query(SecurityEvent).all()
    servers = db.query(Server).all()
    policies = db.query(SecurityPolicy).all()
    return templates.TemplateResponse("events.html", {"request": request, "events": events, "servers": servers, "pol

@app.get("/events/create")
def event_create_form(request: Request, db: Session = Depends(get_db)):
    servers = db.query(Server).all()
    policies = db.query(SecurityPolicy).all()
    return templates.TemplateResponse("event_form.html", {"request": request, "servers": servers, "policies": polici

@app.post("/events/create")
def event_create(request: Request, server_id: int = Form(...), policy_id: int = Form(...),
                 event_type: str = Form(...), severity: str = Form(...), description: str = Form(""),
                 db: Session = Depends(get_db)):
    event = SecurityEvent(server_id=server_id, policy_id=policy_id, event_type=event_type,
                          severity=severity, description=description)
    db.add(event)
    db.commit()
```

Рисунок 2 - Исходный код основных маршрутов приложения часть 2.

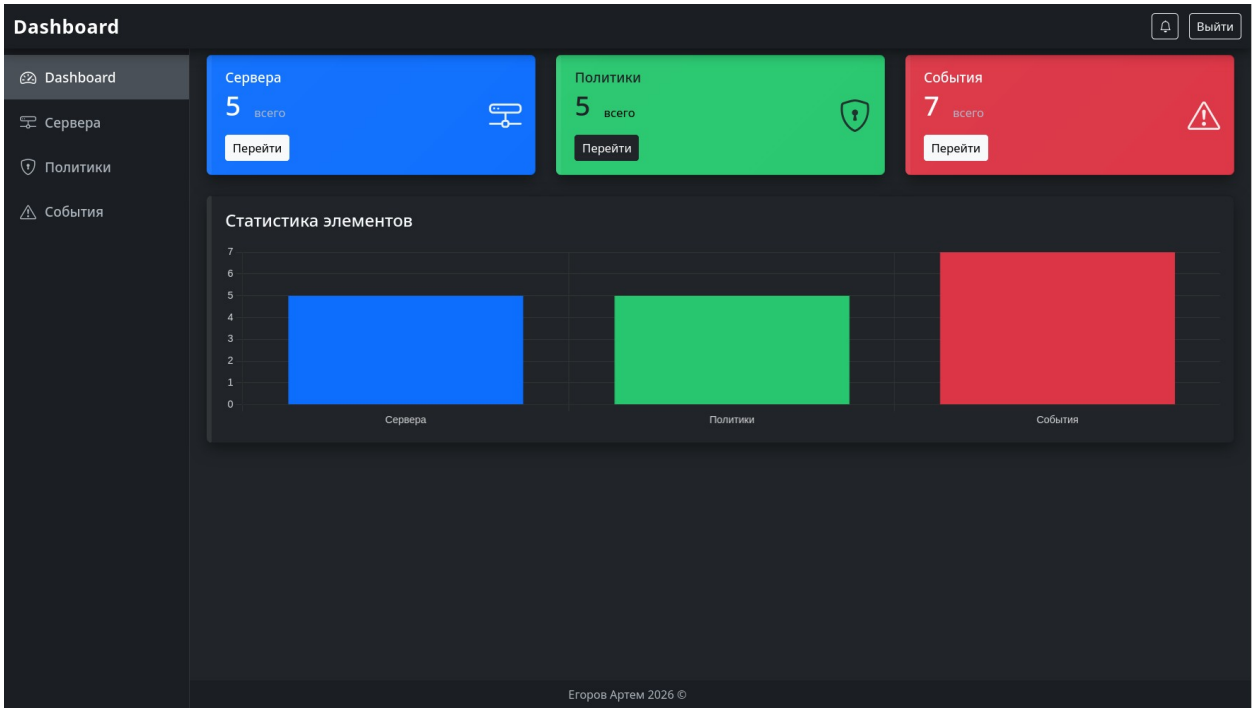

Рисунок 3 - Пример интерфейса панели мониторинга.

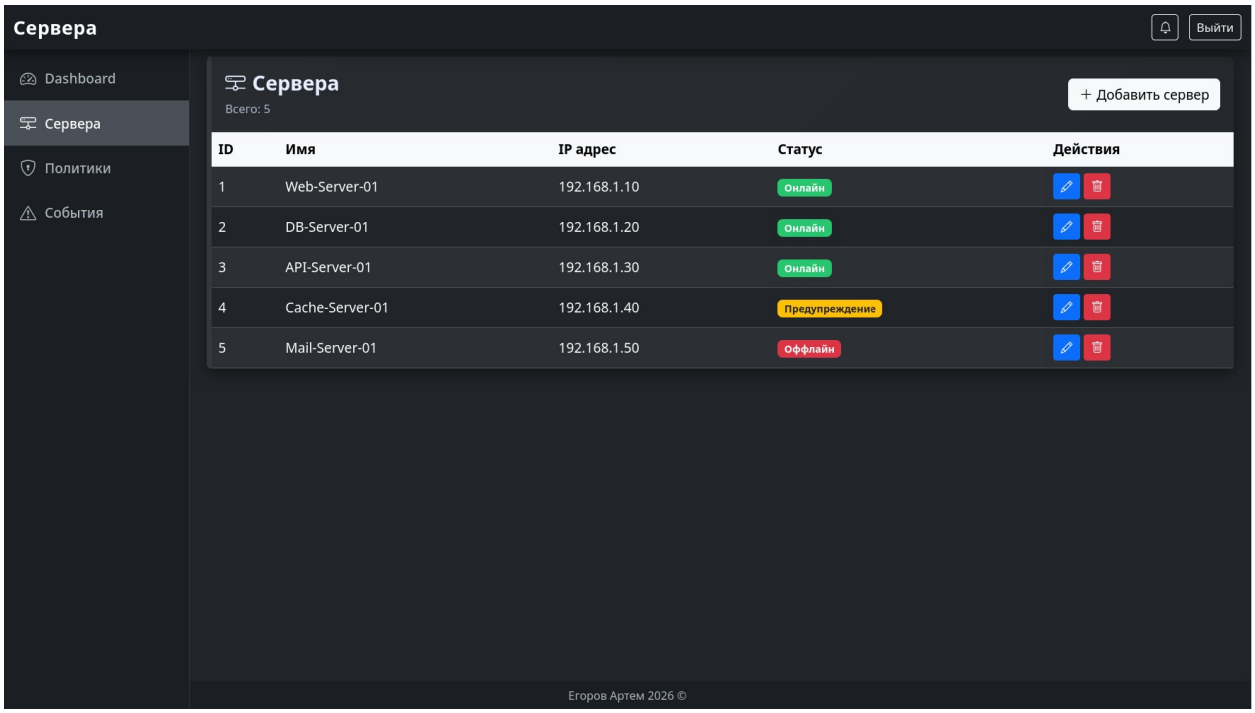

Рисунок 4 - Пример интерфейса панели мониторинга.

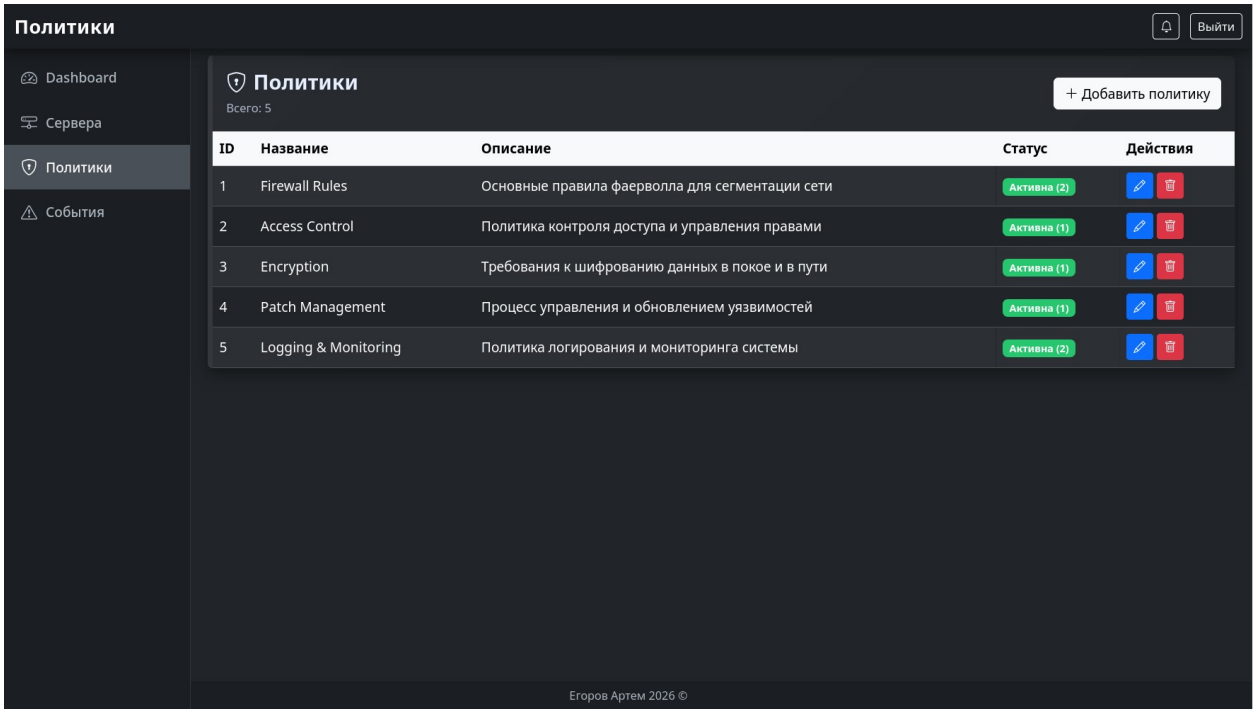

Рисунок 5 - Раздел управления политиками безопасности.

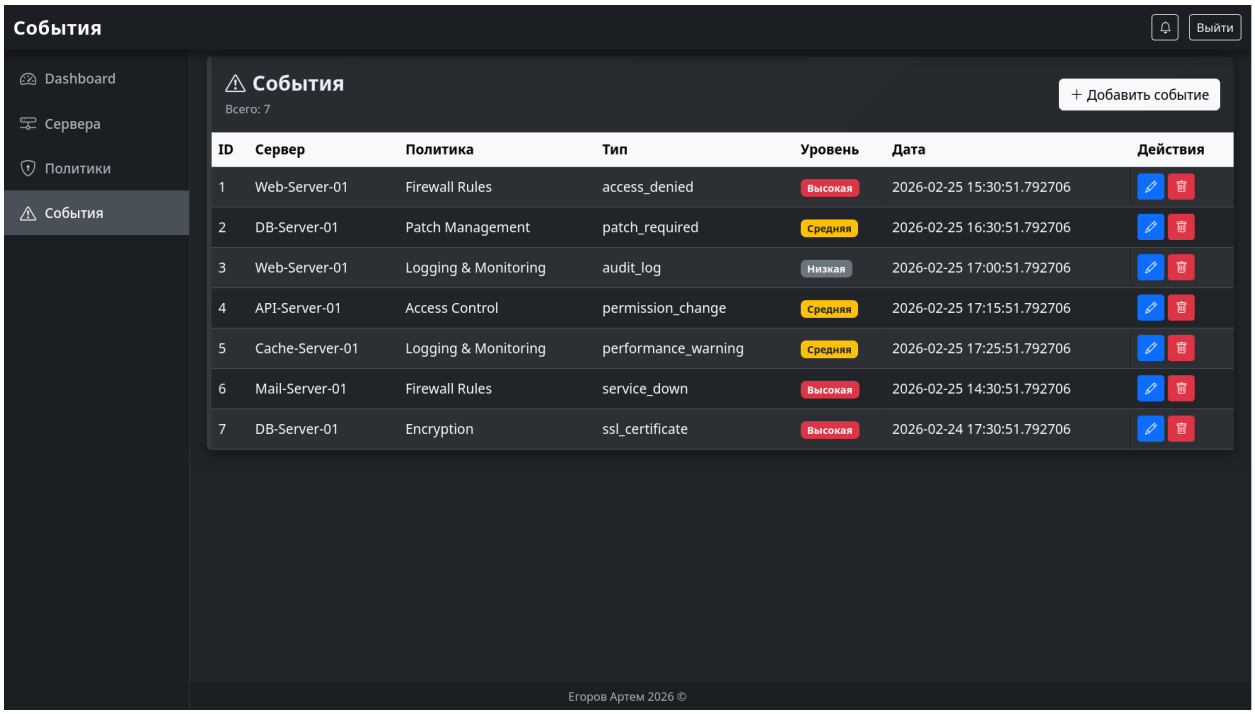

Рисунок 6 - Страница логирования событий безопасности.