

Разработка серверной части приложения для учёта и мониторинга оборудования предприятия минерально-сырьевого комплекса

Автор проекта: Очеповский Данила Дмитриевич
Желаемая обр. программа магистратуры: 09.04.04 Программная инженерия –
Системное и прикладное программное обеспечение

Цель

Повышение эффективности рабочей деятельности сотрудников за счёт обеспечения эффективного учёта оборудования и мониторинга его показателей, упрощение процесса регистрации и отслеживания оборудования, а также предоставление доступа к актуальной информации для всех сотрудников компании.

Задачи

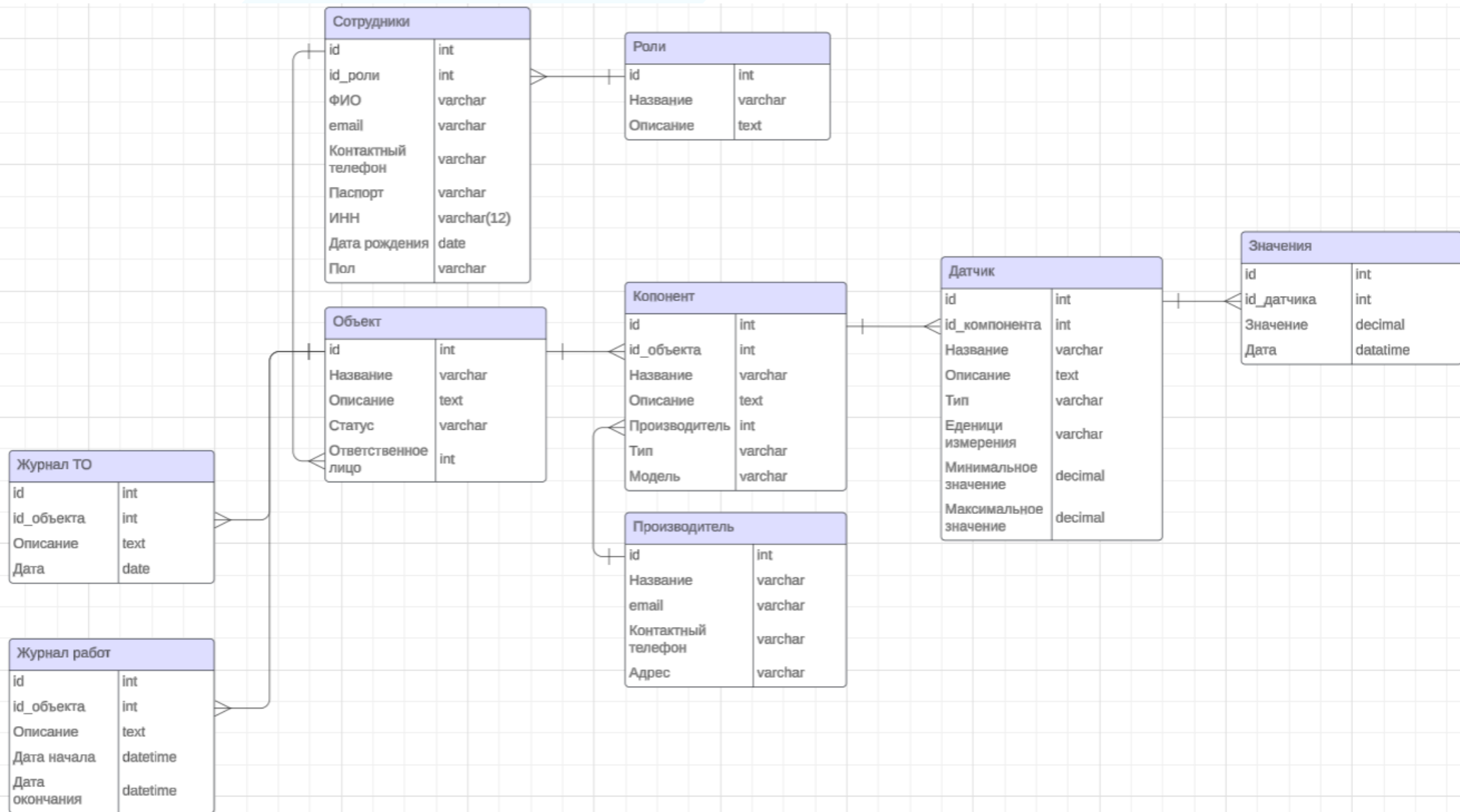
- 1. Анализ требований пользователей и функциональных возможностей системы мониторинга.**
- 2. Выбор технологий и разработка архитектуры, включающая в себя проектирование базы данных.**
- 3. Разработка основного функционала приложения, включая сбор, обработку и представление данных.**

Описание проекта

Разработанное приложение предоставляет следующий функционал:

- Регистрация и авторизация сотрудников на сайте;
- Возможность добавления, изменения и удаления объектов минерально-сырьевого комплекса;
- Возможность добавления, изменения и удаления компонентов, которые установлены на объектах предприятия;
- Возможность добавления, изменения и удаления датчиков для получения аналитики и обратной связи, установленных на компонентах;
- Возможность сбора и анализа значений, получаемых с датчиков;
- Возможность фильтрации, сортировки и группировки данных, получаемых с датчиков;
- Возможность добавления данных о техобслуживании компонентов и датчиков;
- Возможность хранения и учета данных о производителях компонентов и датчиков.

Датологическая модель БД



Разработка серверной части

```
from sqlalchemy import Column, Integer, VARCHAR, TEXT, ForeignKey

from app.database import Base

class Object(Base):
    __tablename__ = "object"

    id = Column(Integer, primary_key=True)
    name = Column(VARCHAR, nullable=False)
    description = Column(TEXT)
    status = Column(VARCHAR, nullable=False)
    responsible_person_id = Column(Integer, ForeignKey("users.id"))
```

Пример кода(создание модели объекта)

```
from pydantic import BaseModel
```

```
class SObject(BaseModel):
    name: str
    description: str
    status: str

    class Config:
        orm_mode = True
```

```
class SObjectEdit(BaseModel):
    name: str
    description: str
    status: str
    responsible_person_id: int

    class Config:
        orm_mode = True
```

Пример кода(классы создания и редактирования объекта, валидация данных)

Логирование

```
5 \\"manufacturer\\".\n[SQL: INSERT INTO components (name, description, type, model, object_id, manufacturer_id) VALUES ($1::VARCHAR, $2::VARCHAR, $3::VARCHAR, $4::VARCHAR, $5::INTEGER, $6::INTEGER) RETURNING components.id]\n[parameters: ('string', 'string', 'string', 'string', 0, 0)]\n(Background on this error at: https://sqlalche.me/e/20/gkpj)"
, "table": "components"}
```

INFO: 127.0.0.1:53544 - "POST /components/add HTTP/1.1" 500 Internal Server Error

Вывод логов в консоль

Авторизация

Были реализованы функции для автоматического извлечения токена из cookies-файлов, получения пользователя и проверки его прав доступа.





Данный функционал используется, например, для получения объектов текущего пользователя:

```
def get_token(request: Request):
    token = request.cookies.get("my_access_token")
    if not token:
        raise TokenAbsentException
    return token

async def get_current_user(token: str = Depends(get_token)):
    try:
        payload = jwt.decode(
            token, settings.SECRET_KEY, settings.ALGORITHM
        )
    except ExpiredSignatureError:
        raise TokenExpiredException
    except JWTError:
        raise IncorrectTokenFormatException
    user_id: str = payload.get("sub")
    if not user_id:
        raise UserIsNotPresentException
    user = await UserDao.find_one_or_none(id=int(user_id))
    if not user:
        raise UserIsNotPresentException

    return user

async def get_current_admin_user(current_user: Users = Depends(get_current_user)):
    if current_user.role_id != 1:
        raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED)
    return current_user
```

Загрузить мои объекты	
Объект 1 Описание объекта 1 Статус  	Тестовый объект Описание тестового объекта Статус  

```
[
  {
    "id": 3,
    "name": "Объект 1",
    "description": "Описание объекта 1",
    "status": "Активен",
    "responsible_person_id": 1
  },
  {
    "id": 2,
    "name": "Тестовый объект",
    "description": "Описание тестового объекта",
    "status": "Не действителен",
    "responsible_person_id": 1
  }
]
```

Оценка и обоснование оригинальности предлагаемого решения

Большинство решений, имеющихсся на рынке, основываются на методе диагностики по состоянию. Веб-приложение предоставляет доступ к обширным данным о состоянии оборудования, параметрах технологических процессов и других важных показателях. Анализ этих данных позволяет предсказывать отказы, принимать более обоснованные и точные решения, оптимизировать планирование и управление ресурсами, что также увеличивает общую эффективность производства и конкурентоспособность компании;

Несмотря на то, что приложение разрабатывалась под задачи минерально-сырьевого комплекса, его структура и перечень используемых инструментов позволяют легко масштабировать и модифицировать его под задачи практически любого современного предприятия.

Перечень использованных в проекте технологий

База данных



Backend



SQLAlchemy

Контроль версий



Соответствие результатов оцениваемым показателям

В ходе выполнения данного проекта разработана серверная часть веб-приложение для учёта и мониторинга оборудования предприятия минерально-сырьевого комплекса.

В процессе разработки выполнены следующие задачи:

1. Выявлены основные потребности компании.
2. Спроектирована архитектура для веб-приложения.
3. Спроектирован API для взаимодействия клиента и сервера

Система разработана с использованием современных технологий, что обеспечивает её высокую производительность и надёжность.

Для того чтобы избежать повторения одного и того же кода был реализован класс BaseDAO служит абстрактным базовым классом для всех DAO (Data Access Object) в проекте.

Был реализован кастомный логгер для эффективной отладки сбоев в работе сервиса.

Ссылка на репозиторий: <https://github.com/OcHeNas/api>