КЭШИРОВАНИЕ С REDIS

ВВЕДЕНИЕ

Задачи урока

01 Научиться кэшировать запросы с использованием Redis

02 Создать бэкенд для хранения кэша с использованием Redis

Кэширование в Redis

это процесс сохранения часто запрашиваемых данных в оперативной памяти Redis для ускорения доступа к этим данным

Когда клиент запрашивает данные, сначала проверяется наличие кэша Redis, и в случае его наличия данные извлекаются из кэша вместо обращения к источнику данных (например, базе данных). Это позволяет значительно снизить временные затраты на выполнение запросов и уменьшить нагрузку на источник данных

Redis обладает высокой скоростью доступа к данным

благодаря хранению данных в памяти, а также поддерживает различные операции с кэшем, такие как чтение, запись, обновление и удаление данных

Redis предлагает возможности для установки времени жизни кэша

поддерживает различные стратегии кэширования, позволяющие выбирать оптимальные способы хранения и обновления данных в кэше

RedisBackend

это бэкенд для хранения сессий во фреймворке FastAPI, который использует Redis в качестве хранилища данных для сессий

Redis предоставляет высокую скорость доступа к данным

благодаря использованию оперативной памяти, что делает его идеальным выбором для хранения сессий, особенно в высоконагруженных приложениях

Бэкенд RedisBackend предоставляет интерфейс для чтения, записи и удаления сессий в Redis, обеспечивая надежное и эффективное управление сеансами пользователей в FastAPI-приложении

Он также поддерживает дополнительные функции, такие как установка времени жизни сессии и возможность использования расширенных возможностей Redis для более гибкого управления сессиями

RedisBackend является популярным выбором для хранения сессий в FastAPI, особенно при работе с масштабируемыми и высоконагруженными приложениями

Для начала зададим, где будем хранить наш кэш

```
...
@app.on_event("startup")
async def startup_event():
    redis = aioredis.from_url("redis://localhost", encoding="utf8",
decode_responses=True)
    FastAPICache.init(RedisBackend(redis), prefix="fastapi-cache")
```

Далее зададим правила кэша для нашего эндпоинта

```
...
@router.get('/')
@cache(expire=30)
def get_all():
    session = SessionBuilder(
        Connection(
            server='localhost',
            port=5432,
            user='postgres',
            password='password',
            db_name='synergy',
            sql_type='PostgresSQL'
        )
    )
    session = session.buid()
    result = session.query(Users).all()
    time.sleep(3)
    return result
```

Декоратор @cache(expire=30)

инструмент для кэширования функций в Python с использованием времени жизни кэша 30 секунд

Когда функция вызывается с определенным набором аргументов, результат ее выполнения сохраняется в кэше. При последующих вызовах с теми же аргументами функция не выполняется повторно, а данные извлекаются непосредственно из кэша, что значительно ускоряет процесс. Установка времени жизни кэша позволяет автоматически устаревать сохраненные данные и перезапускать функцию для получения новых результатов

ИТОГИ

- Для создания бэкэнда где будет храниться кэш используем RedisBackend
- Если наши данные обновляются редко, то выставляем нужное время жизни нашего кэша в декораторе @cache(expire=30)