ТЕСТОВОЕ ЗАДАНИЕ:
В ответе необходимо вписать ссылку на GitHub.

Мы рады давать возможность попробовать себя кандидатам даже без опыта, однако именно поэтому у нас очень много желающих. К сожалению, у нас нет физической возможности проводить индивидуальные собеседования с каждым кандидатом, поэтому просим откликаться только тех, кто действительно готов выполнить тестовое задание.
Благодарим за понимание!

Если у вас возникнут вопросы по заданию или вы по какой то причине не смогли оставить ссылку на выполненное тестовое задание в нашей форме, пишите сюда ТГ @Hobrus

Техническое задание (ТЗ) на разработку сервиса «Одноразовые секреты»

Суть задачи:
Разработать HTTP-сервис на FastAPI, в котором можно хранить конфиденциальные данные (далее — «секреты»).
Секрет обязательно должен выдаваться только один раз: после первого запроса к нему по уникальному ключу он становится недоступным.

Кеширование:
Созданный секрет обязательно должен находиться в серверном кеше не менее 5 минут с момента создания.

Требования к хранению и безопасности
Шифрование: Все «секреты» должны храниться в зашифрованном виде (не в открытом тексте).

Запрет кеширования на клиенте: Необходимо выставлять соответствующие HTTP-заголовки, чтобы запретить любое кеширование на стороне клиента и промежуточных прокси.

Работа с базой данных (PostgreSQL)
В дополнение к кешированию (и независимо от того, где именно хранятся секреты оперативно) сервис должен использовать PostgreSQL для хранения определённых данных.
Рекомендуется (но не ограничивается этим) сохранять следующую информацию:

Логи о действиях в сервисе (например, создание секрета, первое чтение, удаление, время, IP-адреса, дополнительные метаданные и т. д.).

Дополнительные сущности (при необходимости), например, учётные записи пользователей, аудит, статистика использования сервиса и т. п.

Конкретная реализация структуры БД (модели данных) остаётся на усмотрение исполнителя, главное — обеспечить корректную работу логирования и иных требуемых функций, чтобы была возможность расширять сервис в будущем.

Инфраструктурные требования
Контейнеризация: Проект должен разворачиваться и запускаться через Docker (Dockerfile и/или docker-compose).

Необходимо обеспечить автоматическую или полуавтоматическую развёртку сервиса вместе с PostgreSQL (если она не предоставляется как внешний сервис).

Предусмотреть переменные окружения для конфигурации подключения к базе данных (POSTGRES_HOST, POSTGRES_PORT, POSTGRES_DB, POSTGRES_USER, POSTGRES_PASSWORD и т. д.).

Код:

Должен соответствовать требованиям PEP8 и использовать type hints.

Должен быть написан на FastAPI.

Документация:

Обязательна документация по публичному API: это может быть Swagger/OpenAPI (автоматически поддерживается FastAPI) или описание в формате Markdown.

Описание необходимых ручек (эндпоинтов)
1. Создание секрета
POST /secret
Тело запроса (JSON) может содержать:

secret (string) — обязательный параметр, конфиденциальные данные.

passphrase (string) — опциональный параметр, фраза-пароль для дополнительной защиты (например, может потребоваться при удалении).

ttl_seconds (number) — опциональный параметр, время жизни секрета в секундах.

Пример тела запроса:

{
  "secret": "доступ_к_конфиденциальным_данным",
  "passphrase": "my_passphrase",
  "ttl_seconds": 3600
}
Пример ответа (JSON):
{
  "secret_key": "уникальный_идентификатор"
}
После успешного создания секрета сервис должен:

Сохранить секрет.

Залогировать создание секрета в PostgreSQL (например, ID секрета, время создания, IP-адрес, при необходимости — указание ttl_seconds и т. д.).

2. Получение секрета
GET /secret/{secret_key}
При первом запросе по secret_key необходимо вернуть ранее сохранённый «секрет».

После успешного получения «секрета» повторный запрос по тому же secret_key не должен выдавать конфиденциальные данные.

Пример ответа (JSON):
{
  "secret": "доступ_к_конфиденциальным_данным"
}
После успешного возврата «секрета»:
Сохранить в логе (PostgreSQL) факт выдачи секрета (время, IP-адрес и т. д.).

3. Удаление секрета
DELETE /secret/{secret_key}
Может потребоваться, если при создании секрета передавался passphrase (или по иной логике, если это предусмотрено бизнес-требованиями).

При успешном удалении «секрет» становится недоступен даже при первом запросе по secret_key.

Пример ответа (JSON):
{
  "status": "secret_deleted"
}
После успешного удаления:

Сервис должен сделать секрет недоступным даже при первом запросе

Сохранить в логе (PostgreSQL) факт удаления (ключ секрета, время, IP-адрес, метаданные о passphrase и т. д.).

Дополнительные требования
Поддержка времени жизни (TTL):

По истечении заданного срока (ttl_seconds) «секрет» становится недоступен.

Необходимо обеспечить периодическую или событийную очистку просроченных секретов (с учётом кеша).

Не возвращать «секрет» повторно:

После первого прочтения «секрет» не должен более возвращаться.

При удалении по запросу пользователя «секрет» также становится недоступен.

Работа с PostgreSQL:

Логирование основных действий (создание, получение, удаление) должно вестись в PostgreSQL.

При необходимости — хранение доп. информации (связанных сущностей, истории изменений, статистики).

Структура таблиц и детальные поля зависят от реализации. Главное — иметь возможность расширять хранение данных в будущем.

Безопасность:

Сервис должен корректно работать с чувствительными данными, используя шифрование секретов и безопасный доступ к базе.

Запрет кеширования на клиенте через соответствующие заголовки (например, Cache-Control: no-cache, no-store, must-revalidate, Pragma: no-cache, Expires: 0).

Технологический стек:

Язык: Python 3.x

Web-фреймворк: FastAPI

База данных: PostgreSQL

Контейнеризация: Docker (Dockerfile / docker-compose)