Реализовать микросервис на FastAPI для управления пользователями с поддержкой стандартных CRUD-операций и безопасной аутентификацией.

Поля пользователя:

- **id** (целое число, первичный ключ)
- **name** (строка)
- **email** (строка, уникальное)
- **password_hash** (строка, для хранения зашифрованного пароля)
- **created_at** (дата и время создания записи)
- **phone** (строка, опционально)

Требуется реализовать следующие эндпоинты:

1. **GET /users** — Получить список всех пользователей.
2. **POST /users** — Создать нового пользователя (в теле запроса передаются name, email, phone и password; пароль сохраняется только в виде безопасного хеша).
3. **GET /users/{user_id}** — Получить информацию о пользователе по id.
4. **DELETE /users/{user_id}** — Удалить пользователя по id.
5. **PUT /api/user/profile** — Обновить свои личные данные (name, email, phone).
    - Авторизация: HTTP Basic (Authorization: Basic base64(login:password))
    - Менять данные может только владелец своего профиля.
6. **PUT /api/user/password** — Сменить свой пароль.
    - Авторизация: HTTP Basic
    - В теле запроса: новый пароль и его подтверждение
    - Для смены пароля требуется верный текущий логин и пароль; если новые пароли не совпадают — вернуть ошибку.

7. **POST /users/{user_id}/change-password** — Сменить пароль пользователя по id (если уместно, например, для администратора).

Форматы запросов и ответов:

- Пример запроса для обновления профиля (PUT /api/user/profile):


```
{
 "name": "Новое имя",
 "email": "new@email.example",
 "phone": "+7 900 100-20-30"
}
```


- Пример ответа:


```
{
 "result": "ok",
 "user": {
  "name": "Новое имя",
  "email": "new@email.example",
  "phone": "+7 900 100-20-30"
 }
}
```


- Пример запроса для смены пароля (PUT /api/user/password):


```
{
 "new_password": "Qwerty1234",
 "new_password_repeat": "Qwerty1234"
}
```


- Пример ответа:


```
{
 "result": "ok",
 "message": "Пароль успешно изменён"
```

}

- В случае ошибки — соответствующий HTTP статус и сообщение, например:

{

"result": "error",

"message": "Неверные логин или пароль"

}

или

{

"result": "error",

"message": "Пароли не совпадают"

}

Требования:

- Все пароли хранить только в виде безопасных хешей (bcrypt или passlib).
- Для всех операций изменения и удаления пользователь может работать только со своими данными (кроме административных случаев).
- Обрабатывать все ошибки (повторяющийся email, пользователь не найден, неверный пароль, не совпадающие новые пароли, и т.д.) информативно.
- Использовать SQLite как базу данных, без alembic (создание таблиц — через Base.metadata.create_all()).
- Код структурировать по папкам: models/, schemas/, main.py и т.д.
- Для идентификации и авторизации пользователя использовать HTTP Basic авторизацию (логин — email или username, пароль).