

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут» імені Ігоря Сікорського
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 8

З дисципліни: «Компоненти програмної інженерії – 2.
Моделювання та аналіз програмного забезпечення»

Виконав:

студент групи ІТ-73

Старовойтов Руслан Олександрович

Перевірив:

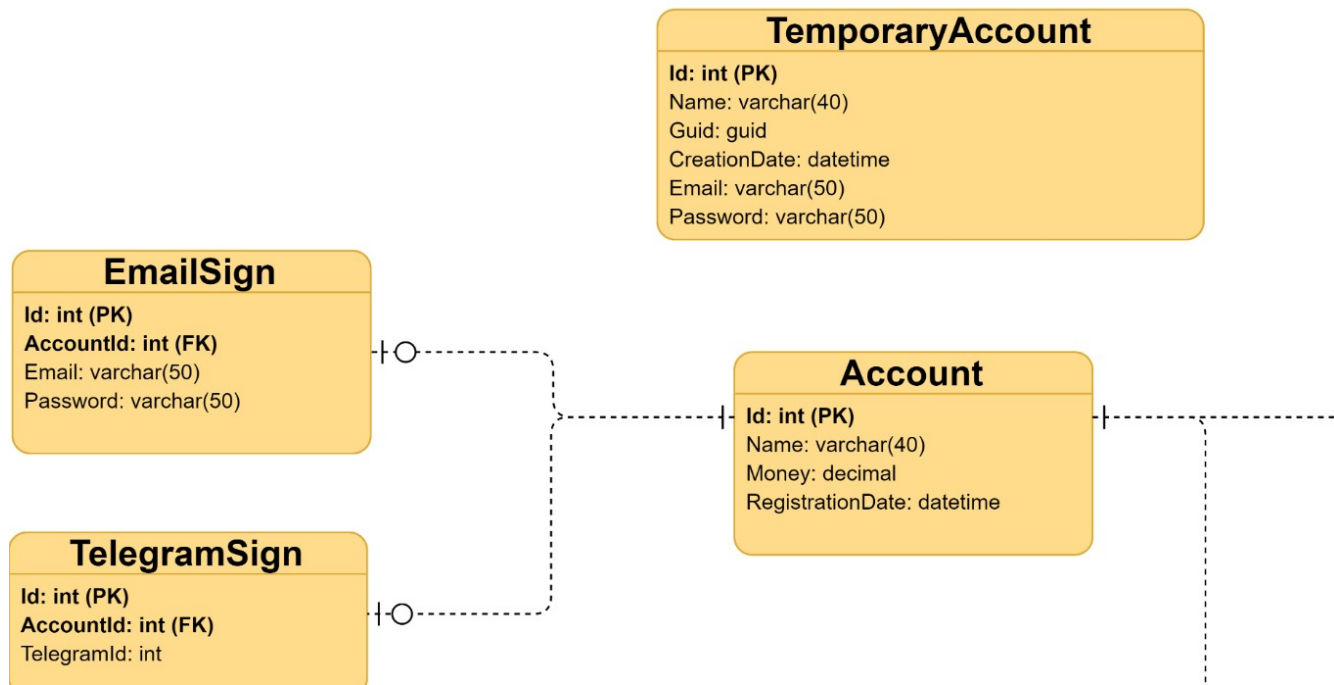
Галушко Д. О

Київ 2020

Тема: Разработка конструктора Telegram-ботов.

Цель: авторизация с помощью виджета.

Для доступа к аккаунту используются следующие таблицы.



На сайте можно войти в через Telegram аккаунт с помощью специальной кнопочки или через email с паролем. Из-за потенциально большого кол-ва способов авторизации через другие сервисы (Google, Facebook, Telegram, GitHub) с помощью OAuth протокола были созданы сущности для разных способов авторизации.

Если пользователь регистрируется через email, то его данные будут помещены в сущность **TemporaryAccount** пока он не подтвердит email.

Теперь подробнее про виджет авторизации телеграма. Вся документация есть на сайте <https://core.telegram.org/widgets/login>. Для использования этого способа авторизации нужно создать бота и сохранить его токен (он нужен для вычисления хеша). На сайте можно выбрать размер виджета. В html страницу авторизации нужно вставить код, который сгенерируется на сайте:

```
<script async
  src="https://telegram.org/js/telegram-widget.js?6"
  data-telegram-login="bots_constructor_bot"
  data-size="large"
  data-auth-url="https://botsconstructor.com/SignIn/LoginWithTelegram"
  data-request-access="write">
</script>
```

После нажатия на кнопку и подтверждения владения аккаунтом будет отправлен Get запрос по указанной ссылке. Далее нужно отсортировать поля и вычислить хэш. Если вычисленный и присланный хеш совпали, то можно авторизовать пользователя.

```
[HttpGet]
public async Task<IActionResult> LoginWithTelegram(string id,
    [FromQuery(Name = "first_name")] string firstName,
    [FromQuery(Name = "last_name")] string lastName,
    [FromQuery(Name = "username")] string username,
    [FromQuery(Name = "photo_url")] string photoUrl,
    [FromQuery(Name = "auth_date")] string authDate,
    [FromQuery(Name = "hash")] string hash)
{
    // @bots_constructor_bot
    string botToken = "913...-Bms8";

    // Эти поля обязательно заполнены
    List<string> authData = new List<string>
    {
        $"id={id}",
        $"auth_date={authDate}"
    };

    // Эти поля могут быть пустыми
    if (firstName != null) authData.Add(item: $"first_name={firstName}");
    if (lastName != null) authData.Add(item: $"last_name={lastName}");
    if (username != null) authData.Add(item: $"username={username}");
    if (photoUrl != null) authData.Add(item: $"photo_url={photoUrl}");

    // Сортировка полей по алфавиту
    authData.Sort();

    // собрать в одну строку
    string dataCheckString = string.Join(separator: "\n", authData);

    // Сравнить хэши
    var authorizationIsValid:bool = IsAuthDataValid(hash, botToken, dataCheckString);

    if (authorizationIsValid)
```

```

1 usage  Ruslan Starovoitov *
private static bool IsAuthDataValid(string hash, string botToken, string dataCheckString)
{
    using (SHA256 mySha256 = SHA256.Create())
    {
        byte[] botTokenByteArray = Encoding.UTF8.GetBytes(botToken);
        byte[] secretKey = mySha256.ComputeHash(botTokenByteArray);
        byte[] allUserData = Encoding.UTF8.GetBytes(dataCheckString);


        using HMACSHA256 hmac = new HMACSHA256(secretKey);
        byte[] myValueByteArray = hmac.ComputeHash(allUserData);
        string calculatedHashString = BitConverter.ToString(myValueByteArray)
            .Replace(OldValue: "-", NewValue: string.Empty);

        if (hash == calculatedHashString.ToLower()) return true;
    }
    return false;
}

```

Вход

➤
Log in as Ruslan



or

✉

🔒

Вход

☐ Запомнить меня

[Забыли пароль?](#)

- Некорректные логин и(или) пароль

Ещё нет своего аккаунта? [Регистрация](#)