Intro

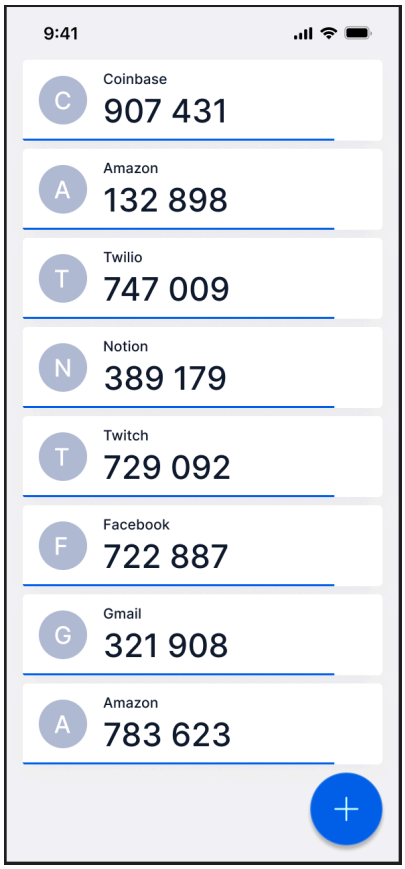
Welcome! We hope you enjoy this exercise.

You will create a 2FA app like Authy, Google Authenticator, among others, to verify **TOTP** (Time-based one-time password) codes, using a provided website for verifying codes. Basically once the user installs the app we'll register that device as a trusted device, by setting up a TOTP factor using Twilio Verify API.

Take-home exercise

Materials provided:

- Sample TOTP application (website): https://verify-totp-1637-ugqh6t.twil.io/index.html
- Twilio Verify TOTP: https://www.twilio.com/docs/verify/quickstarts/totp (only as reference)
- Time-based one-time password (TOTP): https://en.wikipedia.org/wiki/Time-based_one-time_password
- Suggested design



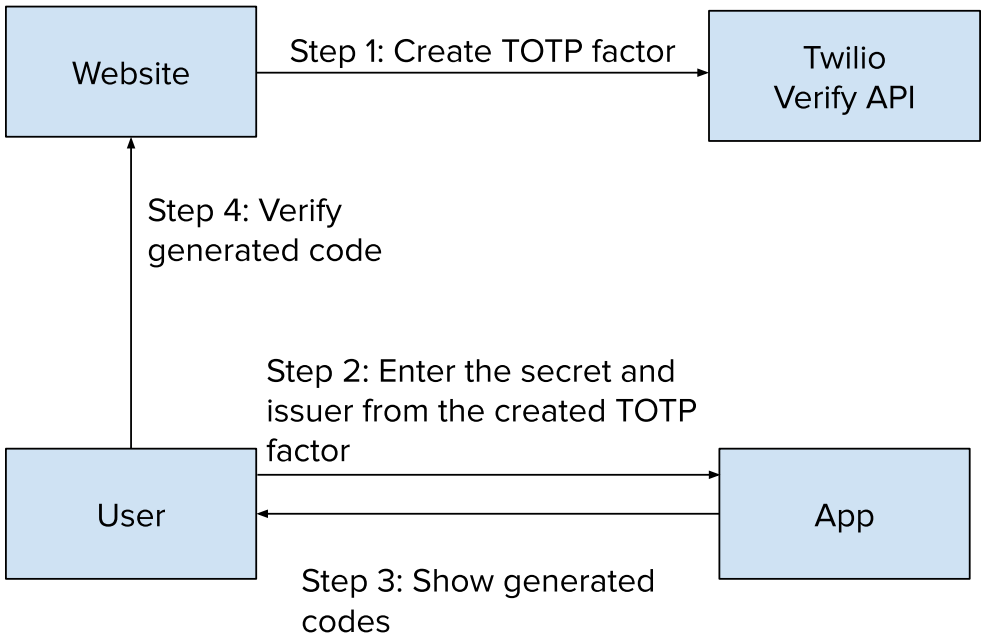
Use case

The use case is the following:

1. The user wants to log in to the website
2. The user enrolls a TOTP factor and store the factor information in the mobile application
3. The website verifies the user, creating a TOTP challenge for that user in the Verify API
4. The website verifies the challenge against the Verify API, to authenticate the user.

Overview

The solution consists in 4 parts:



Step 1: Register the user in the TOTP application

Step 2: Create a mobile app to store the secret and issuer

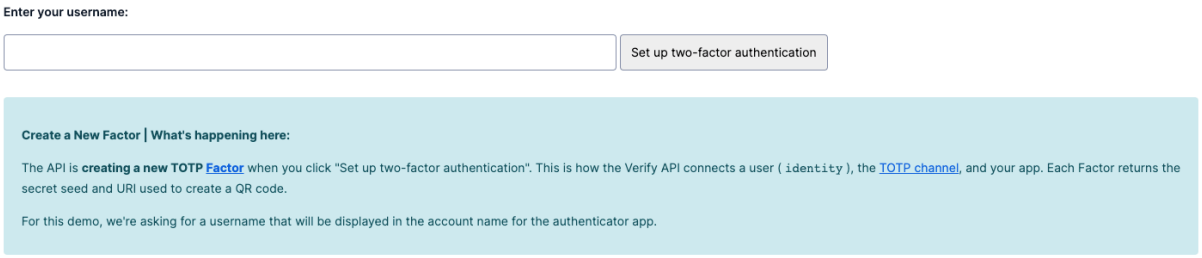
Step 3: Generate codes and show them in the mobile app

Step 4: Verify the generated code in the TOTP application

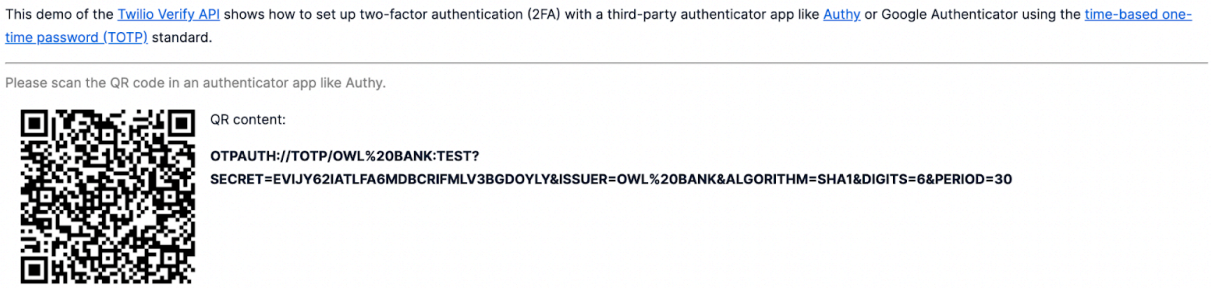
Important Note: We don’t need to code the website.

Step 1: Register the user in the TOTP application

1. Go to the website: https://verify-totp-1637-ugqh6t.twil.io/index.html
2. Enter a username. It should be 3 characters at least



3. Validate that the QR code and QR content are shown



Step 2: Create a mobile app to store the secret and issuer

1. Create a mobile app
2. Add a library to generate TOTP codes
    a. Suggested for Android: https://github.com/marcelkliemannel/kotlin-onetimepassword
    b. Suggested for iOS: https://github.com/lachlanbell/SwiftOTP
3. Pass the secret and issuer to the mobile app using one of the following options:
    a. Scan the QR code to get the information OR
    b. Create a view to type the secret and issuer. You can find the secret and issuer inside the query params of the QR content uri

Note: The secret is already encoded in **Base32**

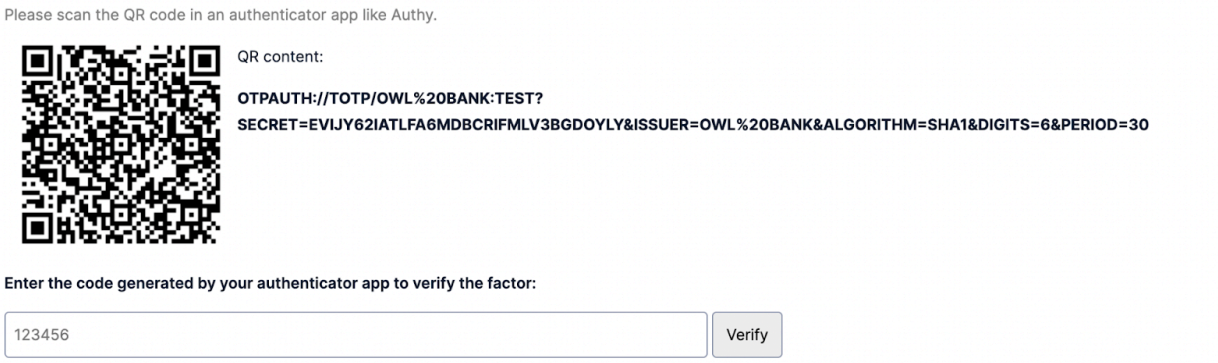
4. Store the information (secret and issuer)

Step 3: Generate codes and show them in the mobile app

1. Generate codes for the secret
    a. Hint: Take into account the secret encoding
2. Show the generated code and issuer, following the suggested design
3. Show a timer indicating the remaining time for the TOTP to be valid, and generate a new one every 30 seconds

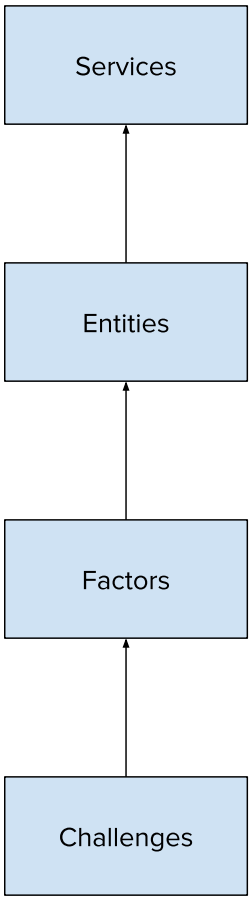
Step 4: Verify the generated code in the TOTP application

1. Enter the generated code associated with the current TOTP factor and click on ‘Verify’



2. Validate that the code is valid

Appendix A



A service is a representation of a customer app, which may have users needing to be authenticated.

An entity is anything that can be authenticated. Typically, a user.

A factor is one of many ways to verify users.
We support multiple factors. For this exercise we'll use the Factor TOTP

All factors have a binding. A binding is a representation of something that belongs to, or is known by the users.
For factor type TOTP, which follows the RFC-6238 algorithm, the Factor contains the seed (Binding.Secret) that is used to generate the TOTP. A Factor contains multiple Challenges.

A challenge is an event of authentication, which has two moments: creation and verification.

To verify a challenge, we send a *AuthPayload* parameter in the body, with whatever the factor needs to be verified.
In the Factor Push, it's going to be a generated code using a TOTP generation algorithm