

# Testing

How the tests of Derafu Auth work, and how to test an application that uses it

Anonymous · 08/10/2026

## Testing

The tests of the package use the real thing wherever it matters: the middlewares of Mezzio, a real SQLite database, and a real Keycloak. The only thing that is not real is where the sessions are kept (a persistence in memory that keeps the contract of the one of PHP), because a test can not have cookies and headers.

```
composer tests
```

**Docker is needed:** the tests of Keycloak start a container. It takes about 20 seconds the first time (the image is downloaded once), and the whole suite about 35.

## What is tested, and against what

| Tests                                                                                                                                     | Against what                                                                                                   |
|-------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| The flow of Keycloak: the login, the callback, the tokens, the keys and their cache, the refresh, the logout, PKCE, the nonce, the roles. | <b>A real Keycloak 26</b> in a container of Docker, with the realm of tests/fixtures/keycloak/test-realm.json. |
| The database provider: the repository, the login, the limit of failed attempts, the rehash, the login page.                               | <b>A real SQLite</b> database, and the real renderer, form and translator.                                     |
| The session: the renewal of its id at login and at logout, the page that is remembered, the redirects, the flash messages.                | The real middlewares of Mezzio (session, flash and authentication), with a session persistence in memory.      |
| The services of the package.                                                                                                              | A container of Symfony that imports the YAML files, with the environment variables.                            |
| The translations: the messages of the code, the templates and the login form.                                                             | The audits of derafu/translation, derafu/twig and derafu/form.                                                 |

## The realm of the tests

The container is `quay.io/keycloak/keycloak:26.8.0` in development mode, with the realm `test`: a confidential client `derafu-auth` that requires PKCE, with the redirect URI `https://app.test/auth/callback` and the post logout redirect URI `https://app.test/bye`; the realm roles `admin` and `editor`; the client role `app-editor`; and the user `ana` (password `secret`, with the roles `admin` and `app-editor`, and the role `view-profile` of the client account, to test that the roles of other clients are not roles of the application). The user and the credentials exist only in the container.

The tests play the browser (`KeycloakBrowser`): they open the authorization URL, send the form of the login of Keycloak and take the redirect to the callback, with the cookies of Keycloak, so its single sign-on works as it does for a user.

## The fixtures

They are in `tests/src/Fixture` (they are not part of the package):

| Fixture                                                    | What it is                                                                                                                                                                                                                            |
|------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>SessionApp</code>                                    | The middlewares of Mezzio around the authentication (session, flash and authentication), with a request builder (path, query, body, headers, session, address) and a way to know the session id that the client has after a response. |
| <code>InMemorySessionPersistence</code>                    | The persistence of the sessions: the id comes in a cookie, a session that was regenerated gets a new id and the old one is destroyed.                                                                                                 |
| <code>UsersDatabase</code>                                 | A SQLite database with users, roles and their hashes.                                                                                                                                                                                 |
| <code>RealKeycloak</code> and <code>KeycloakBrowser</code> | The container of Keycloak, and the user with a browser.                                                                                                                                                                               |
| <code>RecordingHttpClient</code>                           | A PSR-18 client that keeps its requests, to know how many times the keys of the realm were read.                                                                                                                                      |

## Testing an application that uses the package

An application tests its protected pages with the same pieces: the session and the flash middlewares and the authentication middleware of Mezzio in front of its handler, and a session persistence that it controls. The users can be logged in by putting them in the session, as the authentication leaves them (`user` for the database provider, and `user` with `oauth2_token` for Keycloak), so a test does not have to go through Keycloak:

```
$persistence = new InMemorySessionPersistence(); // The one of your tests: see the fixture.  
$persistence->store['a-session'] = [  
    'user' => ['identity' => 'ana@example.com', 'roles' => ['admin'], 'details' =>
```

```
[[]],  
];  
  
$response = $pipeline->handle(  
    $request->withUri(new  
Uri('https://app.test/admin/users'))->withCookieParams(['sid' => 'a-session'])  
);  
  
$this->assertSame(200, $response->getStatusCode());
```

And a request without that session is redirected to the login (or to Keycloak), or answered with a 401 if it is an API:

```
$response = $pipeline->handle($request->withUri(new  
Uri('https://app.test/admin/users')));  
  
$this->assertSame(302, $response->getStatusCode());
```

If the application has a real Keycloak in its tests, it can start the same container with the same realm and add its own client to it.

---

Last updated on 08/10/2026