Paul E. Sevinç, Dr. sc. ETH Zürich

Developing Web Applications: APIs with Play & Scala and GUIs with React & TypeScript

Developing Web Apps

APIs with Play & Scala and GUIs with React & TypeScript

Paul E. Sevinç

This book is for sale at http://leanpub.com/DevWebApps

This version was published on 2024-05-07

Contents

Preface

Welcome to *Developing Web Apps*!

This book shows how to develop a Web application by developing its API with Play & Scala and its GUI with React & TypeScript. Note that the resulting Web application (be it a monolith or a self-contained system[1]) consists of one only deployment unit. This is reflected by there being only one Git/Docker[2] repository.

MVP

This book is about **building the thing right**, not about **building the right thing**. Therefore, we will not dwell on how to come up with a product, let alone worry about how viable the product is. Check out *Product Management in Practice*[a] if you are interested in product management, too.

[a]https://www.oreilly.com/library/view/product-management-in/9781098119720/

Using React with TypeScript for developing GUIs is a no-brainer; I do not have much relevant experience using anything else (other than generating the GUI in the back-end, which is what I did back in 2007 when I implemented Doodle[3] and when JavaScript was turned off in many browsers for security reasons). More importantly, both React and TypeScript are popular, state-of-the-art pieces of technology as the State of JavaScript[4] survey keeps confirming. Using Play with Scala for developing APIs is one of several choices I had; the others were using Play with Java and using Spring Boot with Java (as well as using vanilla Servlets[5] with Java, which is what I did back in 2007 …). From a commercial point of view, I should have chosen Spring Boot (heck, I should have chosen Spring Boot with Kotlin), but I prefer Play over Spring Boot and Scala over both Java and Kotlin.

Enough about me. Let us talk about you for a minute.

[1]https://scs-architecture.org/
[2]Familiarity with Git or Docker is not a prerequisite, but would be helpful for the first part as well as the last part.
[3]https://doodle.com/
[4]https://2022.stateofjs.com/
[5]https://jakarta.ee/specifications/servlet/

Prerequisites

- You need to understand basic Web technologies such as HTML and HTTP. If that is not the case yet, start to *Learn web development*[6] and go back to the MDN project[7] whenever you are unfamiliar with or unsure about a piece of Web technology.
- You need to know Scala to get the most out of the API sections. If you are new to programming, study chapters 1 to 10 as well as 13, 15, and 16 of *Introduction to Programming and Problem Solving Using Scala*[8]. If you are only new to Scala, read *Scala for the Impatient*[9].
- You need to know TypeScript to get the most out of the GUI sections. If you are new to programming, study chapters 1 to 10 as well as 13, 15, and 16 of *Introduction to Programming and Problem Solving Using Scala*[10] (yep, Scala—I do not know of any good introduction to programming using TypeScript yet[11]). If you are only new to JavaScript, read *Modern JavaScript for the Impatient*[12].[13] If you are only new to TypeScript, read the *Get Started* part as well as the *Handbook* part of the TypeScript documentation[14].
- You need to be familiar with Play to get the most out of the API sections. If you are new to Play, take a quick look at its home page[15], skim through the *Getting Started*[16] section, and go through the tutorial[17] in order to gain a first impression of Play.
- You need to be familiar with React to get the most out of the GUI sections. If you are new to React, take a quick look at its home page[18], skim through the *Installation*[19] section, and go through the Quick Start[20] section in order to gain a first impression of React.
- You need to have a Java Development Kit (JDK) installed, OpenJDK[21] for instance.
- You need to have sbt[22] installed.
- You need to have Node.js[23] installed.

[6]https://developer.mozilla.org/en-US/docs/Learn
[7]https://developer.mozilla.org/en-US/
[8]http://www.programmingusingscala.net/home/introduction-to-programming-and-problem-solving-using-scala
[9]https://horstmann.com/scala/index.html
[10]http://www.programmingusingscala.net/home/introduction-to-programming-and-problem-solving-using-scala
[11]I *do* know of an excellent introduction to *Programming with Types* using TypeScript, however.
[12]https://horstmann.com/javascript-impatient/
[13]*«[Y]ou can't learn TypeScript without learning JavaScript!»*
[14]https://www.typescriptlang.org/docs
[15]https://www.playframework.com/
[16]https://www.playframework.com/documentation/latest/Introduction
[17]https://www.playframework.com/documentation/latest/HelloWorldTutorial
[18]https://react.dev/
[19]https://react.dev/learn/installation
[20]https://react.dev/learn
[21]https://adoptium.net/
[22]https://www.scala-sbt.org/
[23]https://nodejs.org/

Under Construction

the remainder of this book as to be (re-)organized and (re-)written

Guided Tours

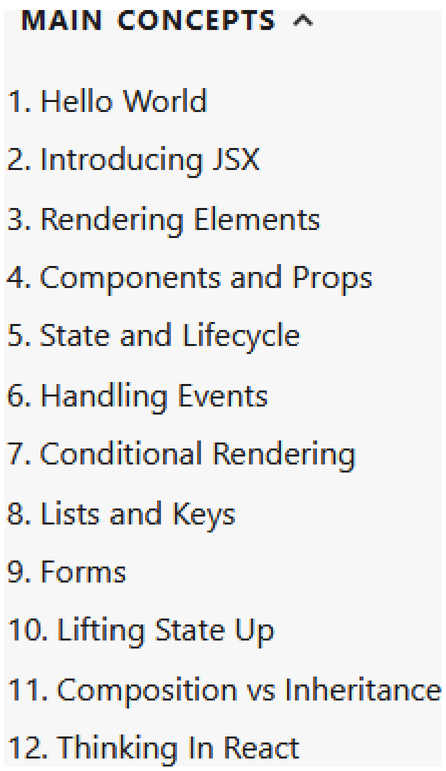
By now you may think that while I am neither teaching TypeScript nor Scala to you, I am at least going to teach React and Play to you. Alas, I have read too many books whose authors have bitten off more than they could chew by trying to explain anything & everything themselves, so I have something else in mind. The parts which go to make the React library and the Play framework are concisely and well explained in the official, freely accessible docs. There is no point in me reinventing the wheel. Instead, my intention is to guide you through the documentation, which allows you to read it lazily[24], and to reinforce what you learn by showing you how every sequence of pages that I reference is applied in a production-level, albeit simple, Web app.

That said, you could read the documentation eagerly[25], before going through the rest of this book. In any case, I strongly recommend (re-)reading the React and Create React App as well as the Play docs (including the *Advanced Guides*, starting at https://reactjs.org/docs/accessibility.html, as well as the *Advanced topics for Scala*[26]) “from cover to cover” eventually, both to solidify your React as well as Play knowledge and to learn of features you did not know existed but that might actually be of use to you.

React

As mentioned in the Prerequisites, if you are new to React, take a quick look at its home page[27], skim through the *Installation*[28] section, and go through the Quick Start[29] section in order to gain a first impression of React’s power and simplicity.

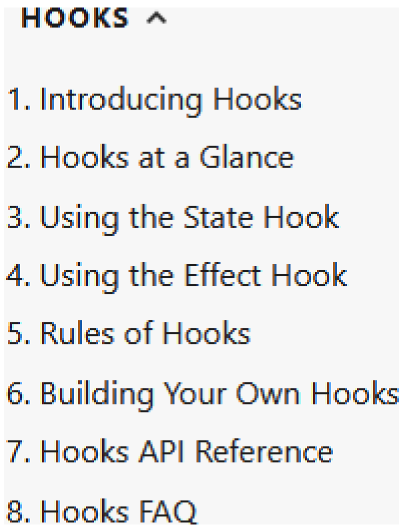
[24]https://en.wikipedia.org/wiki/Lazy_evaluation
[25]https://en.wikipedia.org/wiki/Eager_evaluation
[26]https://www.playframework.com/documentation/latest/ScalaAdvanced
[27]https://reactjs.dev/
[28]https://https://react.dev/learn/installation
[29]https://react.dev/learn


MAIN CONCEPTS ^

1. Hello World
2. Introducing JSX
3. Rendering Elements
4. Components and Props
5. State and Lifecycle
6. Handling Events
7. Conditional Rendering
8. Lists and Keys
9. Forms
10. Lifting State Up
11. Composition vs Inheritance
12. Thinking In React

Main Concepts

I am going to guide you through all twelve *Main Concepts* pages, starting at https://reactjs.org/docs/hello-world.html, and most of the *Hooks* pages, starting at https://reactjs.org/docs/hooks-intro.html.


HOOKS ^

1. Introducing Hooks
2. Hooks at a Glance
3. Using the State Hook
4. Using the Effect Hook
5. Rules of Hooks
6. Building Your Own Hooks
7. Hooks API Reference
8. Hooks FAQ

Hooks

I am also going to extensively reference the Create React App[30] docs, in particular:

[30]https://create-react-app.dev/

- Welcome
  - About Docs[31]
- Getting Started
  - Getting Started[32]
  - Folder Structure[33]
  - Available Scripts[34]
  - ~~Supported Browsers and Features~~
  - Updating to New Releases[35]
- Development
  - Editor Setup[36]
  - ~~Developing Components in Isolation~~
  - ~~Analyzing Bundle Size~~
  - HTTPS in Development[37]
- Styles and Assets
  - Adding Stylesheets[38]
  - ~~Adding CSS Modules~~
  - ~~Adding Sass Stylesheets~~
  - ~~Adding CSS Reset~~
  - ~~Post-Processing CSS~~
  - Adding Images, Fonts, and Files[39]
  - ~~Loading .graphql Files~~
  - Using the Public Folder[40]
  - ~~Code Splitting~~
- Building your App
  - Installing a Dependency[41]
  - Importing a Component[42]
  - Using Global Variables[43]
  - Adding Bootstrap[44]
  - ~~Adding Flow~~
  - Adding TypeScript[45]
  - ~~Adding Relay~~
  - Adding a Router[46]

[31]https://create-react-app.dev/docs/documentation-intro
[32]https://create-react-app.dev/docs/getting-started
[33]https://create-react-app.dev/docs/folder-structure
[34]https://create-react-app.dev/docs/available-scripts
[35]https://create-react-app.dev/docs/updating-to-new-releases
[36]https://create-react-app.dev/docs/setting-up-your-editor
[37]https://create-react-app.dev/docs/using-https-in-development
[38]https://create-react-app.dev/docs/adding-a-stylesheet
[39]https://create-react-app.dev/docs/adding-images-fonts-and-files
[40]https://create-react-app.dev/docs/using-the-public-folder
[41]https://create-react-app.dev/docs/installing-a-dependency
[42]https://create-react-app.dev/docs/importing-a-component
[43]https://create-react-app.dev/docs/using-global-variables
[44]https://create-react-app.dev/docs/adding-bootstrap
[45]https://create-react-app.dev/docs/adding-typescript
[46]https://create-react-app.dev/docs/adding-a-router

- Environment Variables[47]
- ~~Making a Progressive Web App~~
- ~~Measuring Performance~~
- Creating a Production Build[48]

• Testing
- ~~Running Tests~~
- ~~Debugging Tests~~

• Back-End Integration
- Proxying in Development[49]
- Fetching Data[50]
- ~~Integrating with an API~~
- ~~Title & Meta Tags~~

• Deployment
- Deployment[51]

• Advanced Usage
- ~~Custom Templates~~
- ~~Can I Use Decorators?~~
- ~~Pre-Rendering Static HTML~~
- Advanced Configuration[52]
- ~~Alternatives to Ejecting~~

• Support
- ~~Troubleshooting~~

Play

As mentioned in the Prerequisites, if you are new to Play, take a quick look at its home page[53], skim through the *Getting Started*[54] section, and go through the tutorial[55] in order to gain a first impression of Play's power and elegance.

I am going to guide you through the *Main concepts for Scala*, in particular:

- Section introduction[56]
- Configuration API[57]

[47]https://create-react-app.dev/docs/adding-custom-environment-variables
[48]https://create-react-app.dev/docs/production-build
[49]https://create-react-app.dev/docs/proxying-api-requests-in-development
[50]https://create-react-app.dev/docs/fetching-data-with-ajax-requests
[51]https://create-react-app.dev/docs/deployment
[52]https://create-react-app.dev/docs/advanced-configuration
[53]https://www.playframework.com/
[54]https://www.playframework.com/documentation/latest/Introduction
[55]https://www.playframework.com/documentation/latest/HelloWorldTutorial
[56]https://www.playframework.com/documentation/latest/ScalaHome
[57]https://www.playframework.com/documentation/latest/ScalaConfig

- HTTP programming
  - Actions, Controllers and Results[58]
  - HTTP Routing[59]
  - Manipulating HTTP results[60]
  - Session and Flash scopes[61]
  - ~~Body parsers~~
  - ~~Actions composition~~
  - ~~Content negotiation~~
  - Handling errors[62]
- Asynchronous HTTP programming
  - Asynchronous results[63]
  - ~~Streaming HTTP responses~~
  - ~~Comet~~
  - ~~WebSockets~~
- ~~The Twirl template engine~~
- Form submission and validation
  - ~~Handling form submission~~
  - Protecting against Cross Site Request Forgery[64]
  - ~~Custom Validations~~
  - ~~Custom Field Constructors~~
- Working with Json
  - JSON basics[65]
  - JSON with HTTP[66]
  - ~~JSON Reads/Writes/Format Combinators~~
  - JSON automated mapping[67]
  - ~~JSON Transformers~~
- ~~Working with XML~~
- ~~Handling file upload~~
- ~~Accessing an SQL database~~
- ~~Using the Cache~~
- Calling REST APIs with Play WS
  - The Play WS API[68]
  - ~~Connecting to OpenID services~~
  - ~~Accessing resources protected by OAuth~~
- ~~Integrating with Akka~~

[58]https://www.playframework.com/documentation/latest/ScalaActions
[59]https://www.playframework.com/documentation/latest/ScalaRouting
[60]https://www.playframework.com/documentation/latest/ScalaResults
[61]https://www.playframework.com/documentation/latest/ScalaSessionFlash
[62]https://www.playframework.com/documentation/latest/ScalaErrorHandling
[63]https://www.playframework.com/documentation/latest/ScalaAsync
[64]https://www.playframework.com/documentation/latest/ScalaCsrf
[65]https://www.playframework.com/documentation/latest/ScalaJson
[66]https://www.playframework.com/documentation/latest/ScalaJsonHttp
[67]https://www.playframework.com/documentation/latest/ScalaJsonAutomated
[68]https://www.playframework.com/documentation/latest/ScalaWS

- Internationalization with Messages[69]
- Dependency Injection
  - Dependency Injection with Guice[70]
  - ~~Compile Time Dependency Injection~~
- Application Settings
  - Application Settings[71]
  - ~~HTTP request handlers~~
  - ~~Essential Actions~~
  - HTTP filters[72]
- ~~Testing your application~~
- Logging[73]

I am also going to extensively reference the *Common topics*, in particular:

- The build system
  - Contents[74]
  - Overview of the build system[75]
  - About sbt settings[76]
  - Manage application dependencies[77]
  - Working with sub-projects[78]
  - ~~Play enhancer~~
  - ~~Aggregating reverse routers~~
  - ~~Improving Compilation Times~~
  - ~~Cookbook~~
  - ~~Debugging your build~~
- Configuration
  - Configuration[79]
  - Configuration file syntax and features[80]
  - Configuring the application secret[81]
  - Configuring the session cookie[82]
  - ~~Configuring the JDBC connection pool~~
  - ~~Configuring Play's thread pools~~
  - ~~Configuring Akka Http Server Backend~~

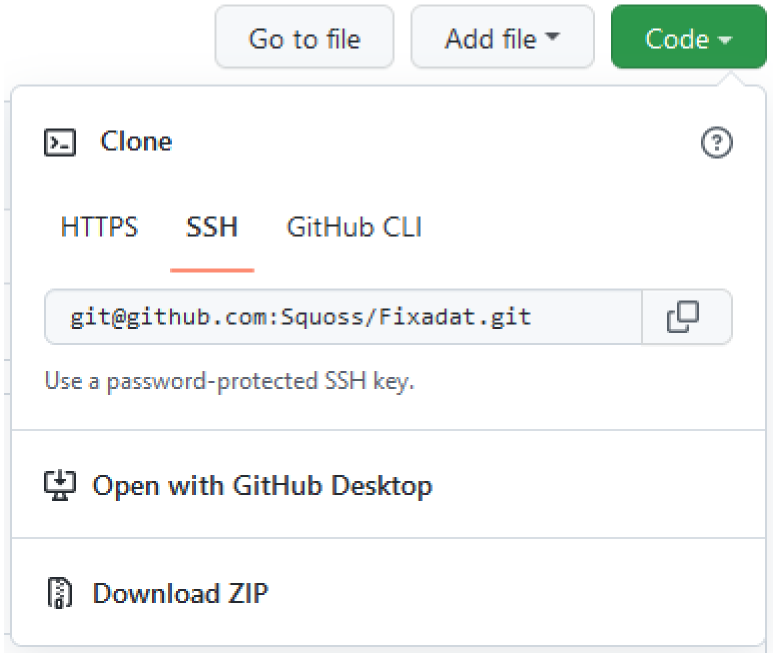
[69]https://www.playframework.com/documentation/latest/ScalaI18N
[70]https://www.playframework.com/documentation/latest/ScalaDependencyInjection
[71]https://www.playframework.com/documentation/latest/ScalaApplication
[72]https://www.playframework.com/documentation/latest/ScalaHttpFilters
[73]https://www.playframework.com/documentation/latest/ScalaLogging
[74]https://www.playframework.com/documentation/latest/Build
[75]https://www.playframework.com/documentation/latest/BuildOverview
[76]https://www.playframework.com/documentation/latest/sbtSettings
[77]https://www.playframework.com/documentation/latest/sbtDependencies
[78]https://www.playframework.com/documentation/latest/sbtSubProjects
[79]https://www.playframework.com/documentation/latest/Configuration
[80]https://www.playframework.com/documentation/latest/ConfigFile
[81]https://www.playframework.com/documentation/latest/ApplicationSecret
[82]https://www.playframework.com/documentation/latest/SettingsSession

  - ~~Configuring Netty Server Backend~~
  - Configuring logging[83]
  - ~~Configuring WS SSL~~
  - ~~Configuring WS Cache~~
- Static assets
  - Static assets[84]
  - Working with public assets[85]
  - ~~Using CoffeeScript~~
  - ~~Using LESS CSS~~
  - ~~Using Sass~~
  - ~~Using JSHint~~
  - ~~Using RequireJs~~
- Built-in HTTP filters
  - Play HTTP filters[86]
  - Configuring gzip encoding[87]
  - Configuring security headers[88]
  - Configuring CORS[89]
  - Configuring CSP[90]
  - Configuring allowed hosts[91]
  - Configuring HTTPS redirect[92]
- ~~Extending Play with modules~~
- ~~Working with Databases~~
- ~~Server Backends~~
- Deploying your application
  - Using Play in production[93]
  - Deploying your application[94]
  - Production configuration[95]
  - ~~Setting up a front end HTTP server~~
  - ~~Configuring HTTPS~~
  - ~~Deploying to a cloud service~~
- ~~Scheduling tasks~~
- ~~Application Shutdown~~
- ~~Integrating with Akka Typed & Cluster Sharding~~

[83]https://www.playframework.com/documentation/latest/SettingsLogger
[84]https://www.playframework.com/documentation/latest/Assets
[85]https://www.playframework.com/documentation/latest/AssetsOverview
[86]https://www.playframework.com/documentation/latest/Filters
[87]https://www.playframework.com/documentation/latest/GzipEncoding
[88]https://www.playframework.com/documentation/latest/SecurityHeaders
[89]https://www.playframework.com/documentation/latest/CorsFilter
[90]https://www.playframework.com/documentation/latest/CspFilter
[91]https://www.playframework.com/documentation/latest/AllowedHostsFilter
[92]https://www.playframework.com/documentation/latest/RedirectHttpsFilter
[93]https://www.playframework.com/documentation/latest/Production
[94]https://www.playframework.com/documentation/latest/Deploying
[95]https://www.playframework.com/documentation/latest/ProductionConfiguration

Schedule like it's 200X

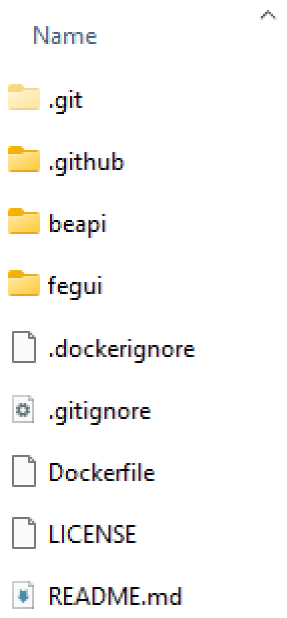
The Web app that serves as a teaching aid is Fixadat. Give it a try at https://fixadat.com/ (and feel free to use it for actually fixing dates & times). Once you got an idea of what Fixadat offers its users, head over to GitHub where its source code is published at https://github.com/Squoss/Fixadat and download the project folder as a ZIP file.



Download ZIP

Extract the ZIP file, which results in a folder called `Fixadat-main`. In that folder, you see that Fixadat consists of a Play subproject (beapi), of a React subproject (fegui), and of a Dockerfile. The `LICENSE` and the `README.md` files are relevant, of course, but not from a technical point of view. The `.git` folder and the `.gitignore` file are by-products of using Git[96] (but only the file being visible when you look at the project structure at https://github.com/Squoss/Fixadat) while the `.github` folder is a by-product of using GitHub Actions[97]. The `.dockerignore` file is a nice-to-have for speeding up a local Docker build.

[96]https://git-scm.com/
[97]https://github.com/features/actions



Project Structure

With your shell, execute the commands `sbt "run -Dconfig.file=conf/insecureLocalhost.conf"` in `Fixadat-main/beapi` and `npm install` as well as `npm start` in `Fixadat-main/fegui`. With your browser[98], visit http://localhost:9000 and http://localhost:3000 to run the apps locally; yes, during development, the API and the GUI are two separate apps. We will see how two become one during deployment.

Access Control

Fixadat requires authorization but does not require authentication. Unless we trust all users to be trustworthy and identify themselves truthfully, how can we achieve one without the other?

"The definitions of trust and trustworthy are often confused. The following example illustrates the difference: if an NSA employee is observed in a toilet stall at Baltimore Washington International airport selling key material to a Chinese diplomat, then (assuming his operation was not authorized) we can describe him as 'trusted but not trustworthy'. Hereafter, we'll use the NSA definition that a trusted system or component is one whose failure can break the security policy, while a trustworthy system or component is one that won't fail."
Ross Anderson, *Security Engineering*[99]

[98]I am using Firefox with the React Developer Tools extension, by the way.
[99]https://www.cl.cam.ac.uk/~rja14/book.html

Identification and Authentication

For the time being, Fixadat does not need any user accounts. If it did, I would not implement them myself anymore (even though I seem to have got them right with respect to password storage[100] years before many major sites thanks to the first edition of Cryptography Engineering[101]). Instead, I would either fall back on so-called social login[102] (and then, with the hope of increasing my users' privacy and reducing my own dependency on GAFAM[103], maybe one made in Germany[104] or FxA[105] if it were available to third parties) or on Identity as a Service (IDaaS).

Authorization

Even though Fixadat does not have any user accounts, we can still control access to resources by taking advantage of capability URLs[106]. In a capability URL such as `https://doodle.com/inturicogmbh`, the capability `inturicogmbh` must not only serve as an identifier and therefore be unique, but also virtually unguessable[107].

We could generate the capabilities ourselves by careful, proper use of a cryptographic pseudorandom number generator[108], or we could leave the heavy lifting to Java's java.util.UUID.randomUUID()[109].

Unfortunately, there is an undeniable risk of exposure[110] with capability URLs. Simply moving the capability to the URLs query string would not mitigate the risk[111]. However, we could move it to the fragment identifier[112] and have the front-end pick it up[113] and provide it to the back-end via a request header.

While we are at it, we can split the capability into a regular identifier (without security properties) and an access token, separating the two concerns: `https://fixadat.com/events/{EVENT_ID}#{ACCESS_TOKEN}`. This has the added benefit of allowing for revoking and re-issuing access tokens as well as for issuing tokens with differing access right.

If you like capabilities for access control, you may also like Macaroons[114].

[100] https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html
[101] https://www.schneier.com/books/cryptography-engineering
[102] https://en.wikipedia.org/wiki/Social_login
[103] https://en.wikipedia.org/wiki/Big_Tech#GAFAM_or_FAAMG
[104] https://www.golem.de/news/single-sign-on-made-in-germany-wettstreit-zwischen-verimi-netid-oder-id4me-1809-136504.html
[105] https://mozilla.github.io/ecosystem-platform/docs/features/firefox-accounts/fxa-overview
[106] https://www.w3.org/TR/capability-urls/
[107] https://www.w3.org/TR/capability-urls/#capability-url-design
[108] https://en.wikipedia.org/wiki/Cryptographically_secure_pseudorandom_number_generator
[109] https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/util/UUID.html#randomUUID()
[110] https://www.w3.org/TR/capability-urls/#risk-of-exposure
[111] https://owasp.org/www-community/vulnerabilities/Information_exposure_through_query_strings_in_url
[112] https://developer.mozilla.org/en-US/docs/Web/API/Location/hash
[113] https://reactrouter.com/web/api/location
[114] https://www.manning.com/books/api-security-in-action

Project Setup

This part consists of seven chapters. Chapters Überproject and Subprojects initiate the project. Chapter Tools of the Trade sets a basic toolbox up. Chapter Two Become One 1 enables the two (sub-)apps to act as one (über-)app during development. Chapter Security as a Forethought hardens the Web app independent of its domain. Chapter So Long, MVC! refactors the Play project in order to nip a Big Ball of Mud in the bud. And chapter Welcome, User-Friendliness! extends the React project such that it can take advantage of both Bootstrap[115] and React Router[116].

Not only the chapter on hardening the Web app, but this entire part is pretty much independent of the domain.

reminder to myself

- replace the Adressabo and Squawg screenshots by Fixadat ones
- review fetchJson.ts
- finish the MVC chapter wrt DI/Guice

[115] https://getbootstrap.com/
[116] https://reactrouter.com/

Überproject

For the most part, we do not really need version control for this book. We do need it to cover delivery & deployment at the end, however. If you do not care about delivery & deployment yet, you could simply create a project folder called `Fixadat` and skip the rest of this chapter:

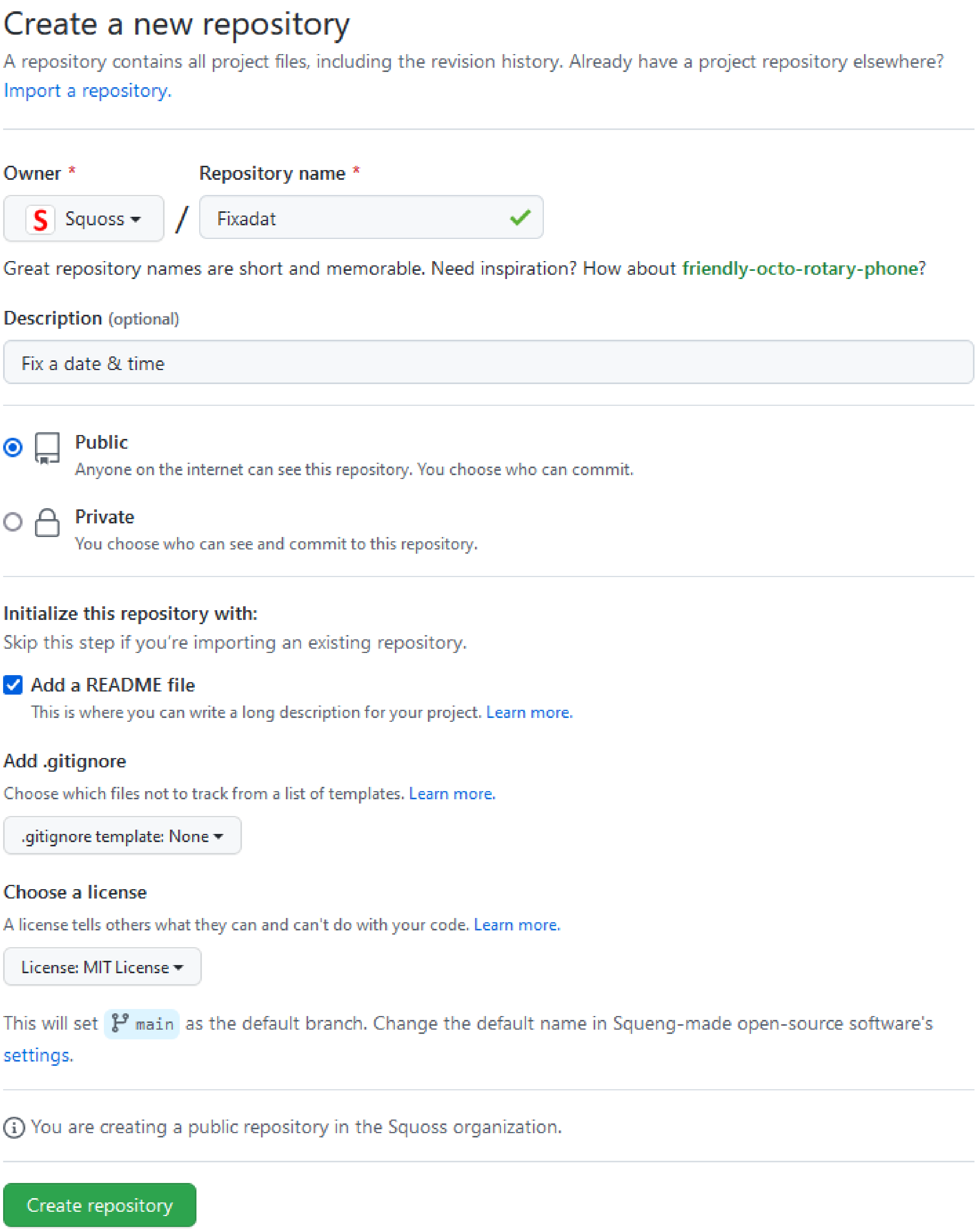
```
mkdir Fixadat
```

Nowadays, thanks to Git[117], there is no reason whatsoever not to use version control anyway. And thanks to (free) services such as GitLab[118], Cloud Source Repositories[119], Bitbucket[120], Azure Repos[121], AWS CodeCommit[122], etc., we do not even have to worry about hosting and backups[123]. We are going to use Microsoft's GitHub[124].

Which came first: The chicken or the egg?

Which came first: The local branch or the remote branch? When starting a brand-new project, I prefer creating its repository remotely and cloning it locally. Creating a repository on GitHub is easy:

[117] https://git-scm.com/
[118] https://gitlab.com
[119] https://cloud.google.com/source-repositories
[120] https://bitbucket.org/
[121] https://azure.microsoft.com/en-us/services/devops/repos/
[122] https://aws.amazon.com/codecommit/
[123] Well, at least in theory, but maybe not in practice.
[124] https://github.com/



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * Squoss / Repository name * Fixadat

Great repository names are short and memorable. Need inspiration? How about friendly-octo-rotary-phone?

Description (optional): Fix a date & time

[✓ Public] Anyone on the internet can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

[✓] Add a README file
This is where you can write a long description for your project. Learn more.

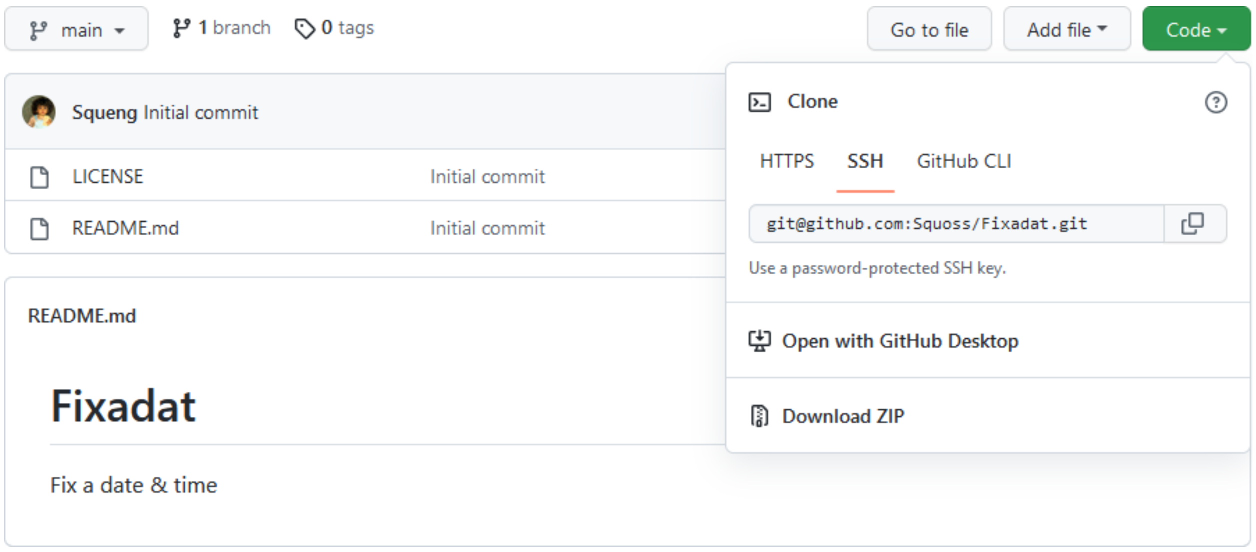
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: MIT License

This will set `main` as the default branch. Change the default name in Squeng-made open-source software's settings.

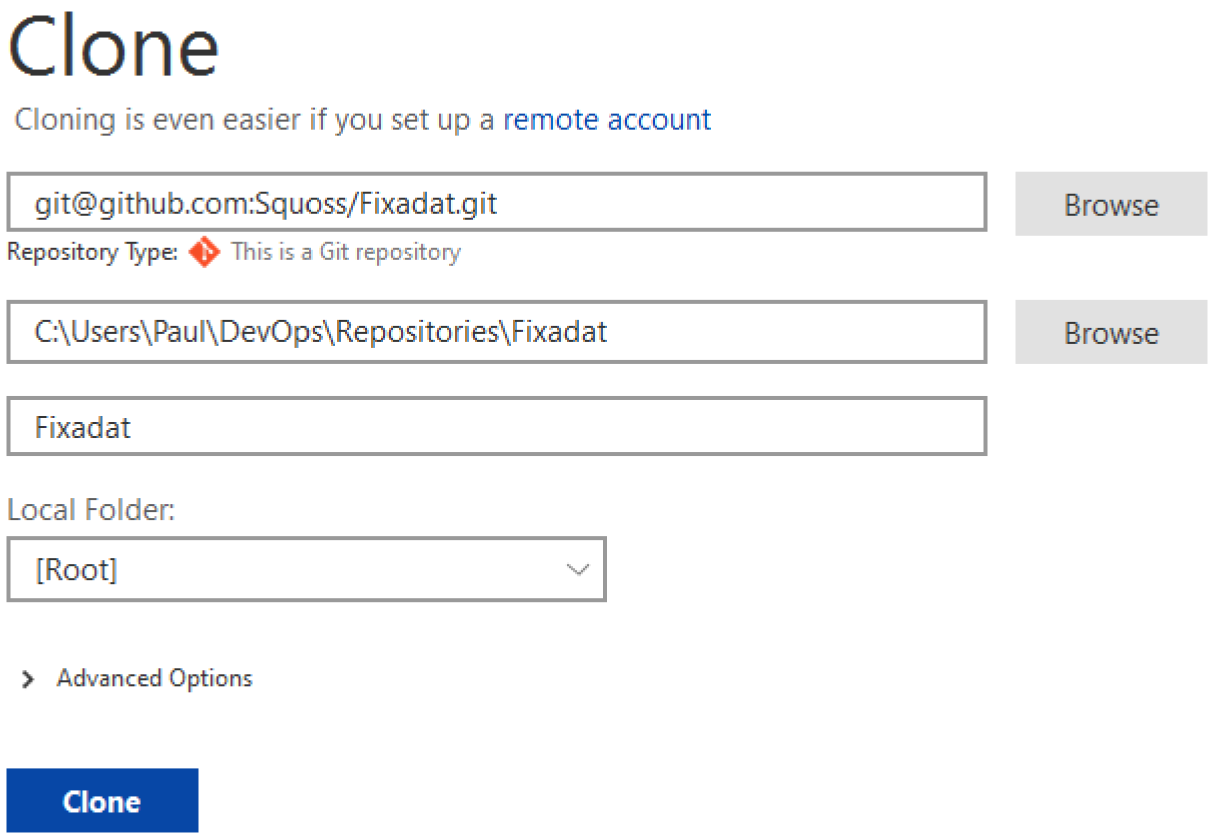
You are creating a public repository in the Squoss organization.

Creating a GitHub repository


main
1 branch
0 tags
Go to file
Add file
Code
Squeng Initial commit
LICENSE Initial commit
README.md Initial commit
README.md
Fixadat
Fix a date & time
Clone
HTTPS SSH GitHub CLI
git@github.com:Squoss/Fixadat.git
Use a password-protected SSH key.
Open with GitHub Desktop
Download ZIP

Created a GitHub repository

So is cloning it locally with Sourcetree[125]:


Clone
Cloning is even easier if you set up a remote account
git@github.com:Squoss/Fixadat.git
Browse
Repository Type: This is a Git repository
C:\Users\Paul\DevOps\Repositories\Fixadat
Browse
Fixadat
Local Folder:
[Root]
Advanced Options
Clone

Cloning a repository

There is now a folder called `Fixadat` that we are going to work in.

[125] https://www.sourcetreeapp.com/

Subprojects

I am going to use `fegui` for the name of the front-end/GUI project and `beapi` for the name of the back-end/API project in an attempt to make them less generic than simply `frontend` or `gui` and `backend` or `api`, respectively, and therefore easily identifiable as well as searchable & replaceable if need be.

React & TypeScript

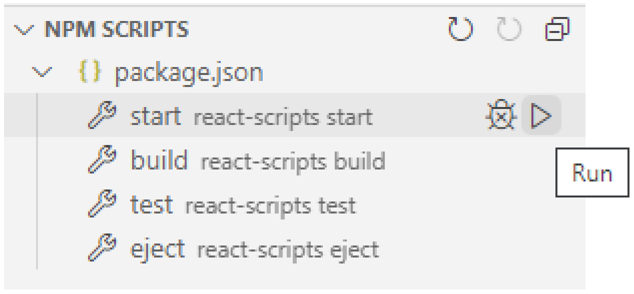
If you have not done so yet, read the following pages (I assume that you already have taken a quick look at React's home page[126], skimmed through the *Installation*[127] section, and gone through the Quick Start[128] section):

- Getting Started[129]
- Folder Structure[130]
- Available Scripts[131]
- Adding TypeScript[132]
- Static Type Checking (with TypeScript)[133]

Since *"Create React App [...] is the best way to start building a new single-page application in React."*[134], that is what we will use to create a brand-new React project with TypeScript support from the very beginning. With your shell, execute the command `npx create-react-app fegui --template typescript` in `Fixadat`. Note that instead of js or jsx[135], the file endings are ts and tsx, respectively, in our case.

With your shell, execute `npm start` in `Fixadat/fegui`. If you are also using VSC, you could run start instead. In both cases, a new browser window or tab should open and display the initial GUI.

[126]https://reactjs.dev/
[127]https://https://react.dev/learn/installation
[128]https://react.dev/learn
[129]https://create-react-app.dev/docs/getting-started
[130]https://create-react-app.dev/docs/folder-structure
[131]https://create-react-app.dev/docs/available-scripts
[132]https://create-react-app.dev/docs/adding-typescript
[133]https://reactjs.org/docs/static-type-checking.html#typescript
[134]https://reactjs.org/docs/create-a-new-react-app.html#create-react-app
[135]https://create-react-app.dev/docs/folder-structure


NPM SCRIPTS
{} package.json
start react-scripts start
build react-scripts build
test react-scripts test
eject react-scripts eject
Run

npm Scripts section in VSC's explorer

Performance as an Afterthought

Even though performance matters, we are not going to measure it[136] in this book. Therefore, we can simplify the project a little bit. Delete `Fixadat/fegui/reportWebVitals.ts` as well as all references to it in `Fixadat/fegui/index.tsx` and then, with your shell, execute the command `npm uninstall web-vitals` in `Fixadat/fegui`.

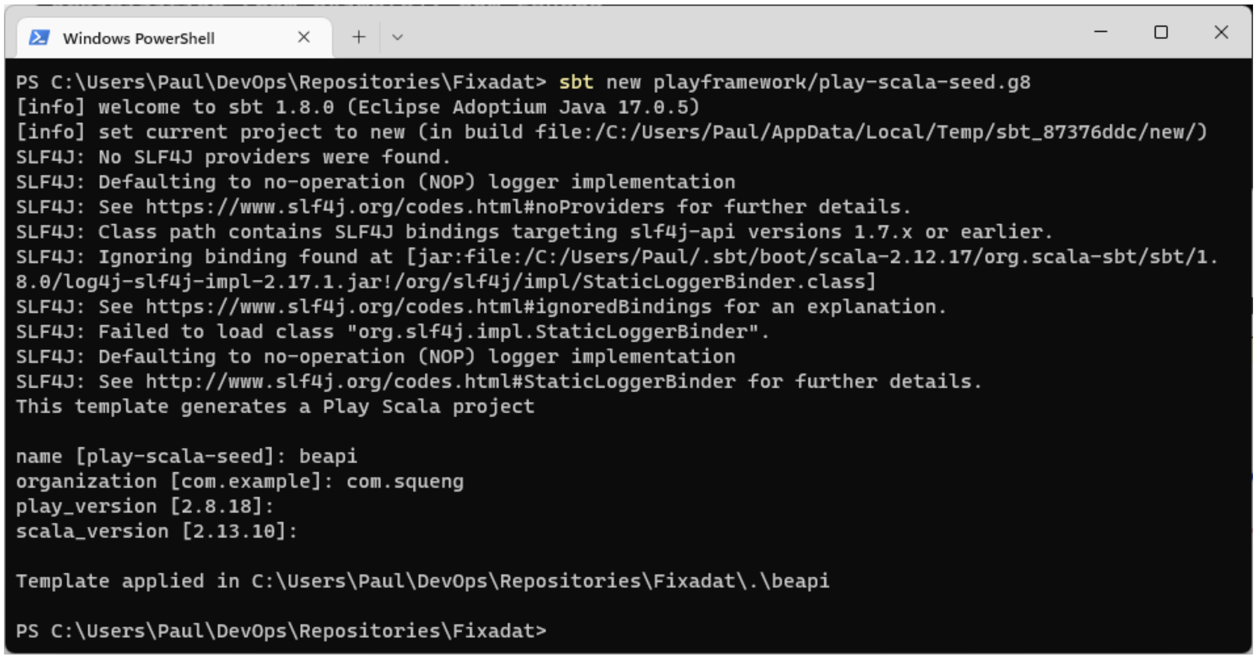
Play & Scala

If you have not done so yet, read the following page (I assume that you already have taken a quick look at Play's home page[137], skimmed through the *Getting Started*[138] section, and gone through the tutorial[139]):

- New to Play[140]

There are several ways to create a Play project. One is to execute the command `sbt new playframework/play-scala-seed.g8` in `Fixadat` to create a brand-new Play project with your shell. When asked for a name and for an organization (in reverse domain name notation[141]), enter `beapi` and whatever reverse domain makes sense. Note that we are going to restructure the anatomy of our Play application[142] by bidding MVC farewell once we have tooled up.

[136] https://create-react-app.dev/docs/measuring-performance/
[137] https://www.playframework.com/
[138] https://www.playframework.com/documentation/latest/Introduction
[139] https://www.playframework.com/documentation/latest/HelloWorldTutorial
[140] https://www.playframework.com/getting-started
[141] https://en.wikipedia.org/wiki/Reverse_domain_name_notation
[142] https://www.playframework.com/documentation/latest/Anatomy



```
PS C:\Users\Paul\DevOps\Repositories\Fixadat> sbt new playframework/play-scala-seed.g8
[info] welcome to sbt 1.8.0 (Eclipse Adoptium Java 17.0.5)
[info] set current project to new (in build file:/C:/Users/Paul/AppData/Local/Temp/sbt_87376ddc/new/)
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
SLF4J: Class path contains SLF4J bindings targeting slf4j-api versions 1.7.x or earlier.
SLF4J: Ignoring binding found at [jar:file:/C:/Users/Paul/.sbt/boot/scala-2.12.17/org.scala-sbt/sbt/1.
8.0/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See https://www.slf4j.org/codes.html#ignoredBindings for an explanation.
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
This template generates a Play Scala project

name [play-scala-seed]: beapi
organization [com.example]: com.squeng
play_version [2.8.18]:
scala_version [2.13.10]:

Template applied in C:\Users\Paul\DevOps\Repositories\Fixadat\.\beapi

PS C:\Users\Paul\DevOps\Repositories\Fixadat>
```

Seeding with Play Scala

In order to quickly check that the Play project was created successfully, execute the command `sbt` in `Fixadat/beapi` and then `run`; note that the second command is executed in sbt's interactive mode in the context of beapi. With your browser, visit http://localhost:9000.

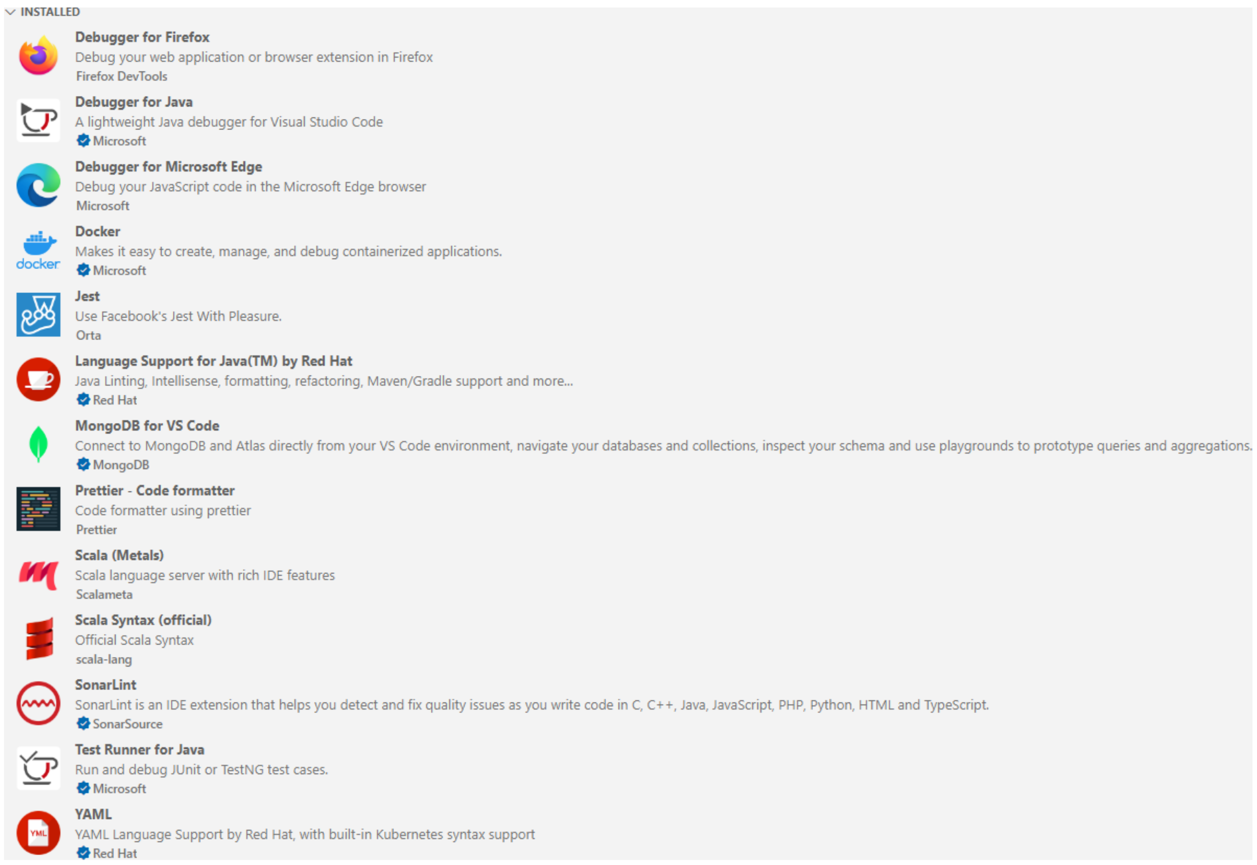
Remove Clutter

Since our frontend is built with React, delete folders `images`, `javascripts`, and `stylesheets` from `Fixadat/beapi/public`. Furthermore, delete folder `.g8` from `Fixadat/beapi` as well as the Giter8[143] plugin from `Fixadat/beapi/project/plugins.sbt`; we will not need them anymore.

[143] http://www.foundweekends.org/giter8/

Tools of the Trade

You need an editor to read and write source code. I am using Visual Studio Code[144] (VSC) with a bunch of extensions, I know quite a few developers who are using IntelliJ IDEA[145] (one friend of mine recommended it to me over twenty years ago), etc.[146]



INSTALLED

Debugger for Firefox
Debug your web application or browser extension in Firefox
Firefox DevTools

Debugger for Java
A lightweight Java debugger for Visual Studio Code
Microsoft

Debugger for Microsoft Edge
Debug your JavaScript code in the Microsoft Edge browser
Microsoft

Docker
Makes it easy to create, manage, and debug containerized applications.
Microsoft

Jest
Use Facebook's Jest With Pleasure.
Orta

Language Support for Java(TM) by Red Hat
Java Linting, Intellisense, formatting, refactoring, Maven/Gradle support and more...
Red Hat

MongoDB for VS Code
Connect to MongoDB and Atlas directly from your VS Code environment, navigate your databases and collections, inspect your schema and use playgrounds to prototype queries and aggregations.
MongoDB

Prettier - Code formatter
Code formatter using prettier
Prettier

Scala (Metals)
Scala language server with rich IDE features
Scalameta

Scala Syntax (official)
Official Scala Syntax
scala-lang

SonarLint
SonarLint is an IDE extension that helps you detect and fix quality issues as you write code in C, C++, Java, JavaScript, PHP, Python, HTML and TypeScript.
SonarSource

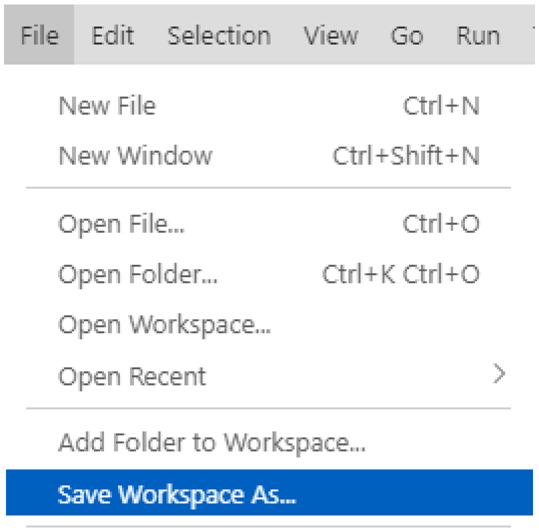
Test Runner for Java
Run and debug JUnit or TestNG test cases.
Microsoft

YAML
YAML Language Support by Red Hat, with built-in Kubernetes syntax support
Red Hat

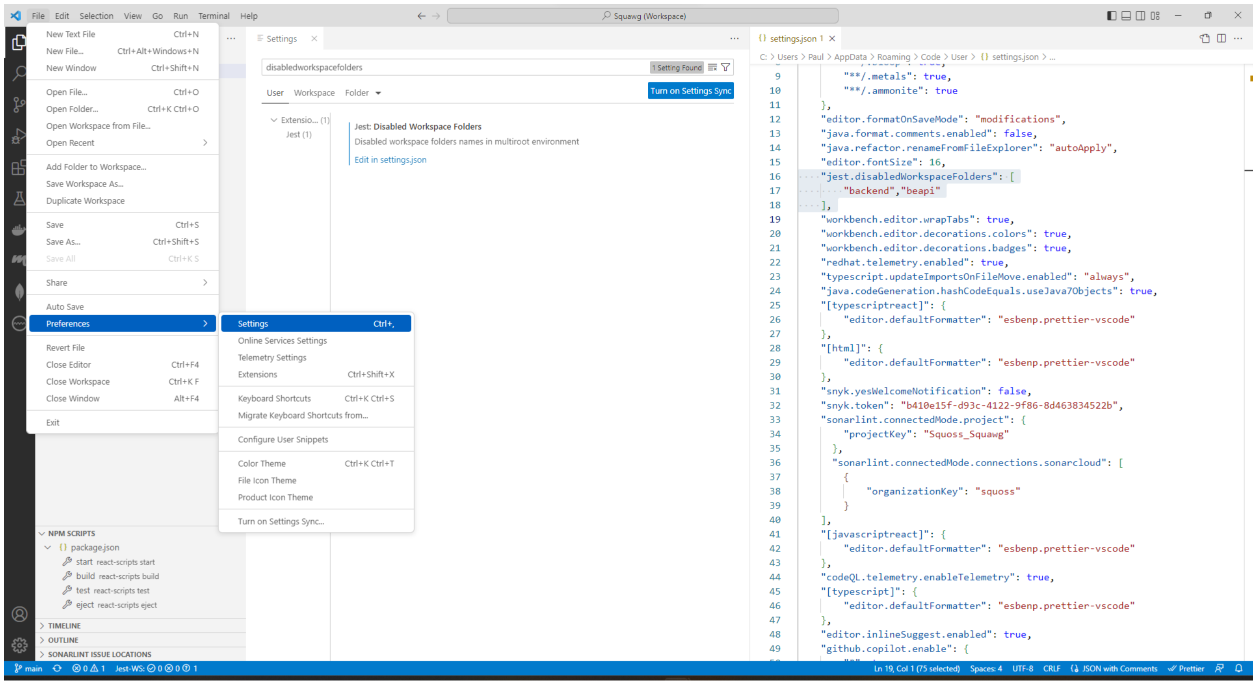
VSC extensions

If you are also using VSC, open a new window. Then *Open Folder…* and choose `beapi` in `Fixadat`. Then *Add Folder to Workspace…* and chosse `fegui` in `Fixadat`. Finally, *Save Workspace As…* so that you do not have to repeat these steps. And if you are also using the Jest extension[147], you should add `beapi` to the disabled workspace folders setting.

[144]https://code.visualstudio.com/
[145]https://www.jetbrains.com/idea/
[146]https://scalameta.org/metals/docs/editors/overview.html
[147]https://marketplace.visualstudio.com/items?itemName=Orta.vscode-jest



VSC folders and workspace



disabledworkspacefolders

Jest: **Disabled Workspace Folders**
Disabled workspace folders names in multiroot environment
Edit in settings.json

C: > Users > Paul > AppData > Roaming > Code > User > {} settings.json > ...

```json
        "**/.metals": true,
        "**/.ammonite": true
    },
    "editor.formatOnSaveMode": "modifications",
    "java.format.comments.enabled": false,
    "java.refactor.renameFromFileExplorer": "autoApply",
    "editor.fontSize": 16,
    "jest.disabledWorkspaceFolders": [
        "backend","beapi"
    ],
    "workbench.editor.wrapTabs": true,
    "workbench.editor.decorations.colors": true,
    "workbench.editor.decorations.badges": true,
    "redhat.telemetry.enabled": true,
    "typescript.updateImportsOnFileMove.enabled": "always",
    "java.codeGeneration.hashCodeEquals.useJava7Objects": true,
    "[typescriptreact]": {
        "editor.defaultFormatter": "esbenp.prettier-vscode"
    },
    "[html]": {
        "editor.defaultFormatter": "esbenp.prettier-vscode"
    },
    "snyk.yesWelcomeNotification": false,
    "snyk.token": "b410e15f-d93c-4122-9f86-8d463834522b",
    "sonarlint.connectedMode.project": {
        "projectKey": "Squoss_Squawg"
      },
      "sonarlint.connectedMode.connections.sonarcloud": [
        {
            "organizationKey": "squoss"
        }
    ],
    "[javascriptreact]": {
        "editor.defaultFormatter": "esbenp.prettier-vscode"
    },
    "codeQL.telemetry.enableTelemetry": true,
    "[typescript]": {
        "editor.defaultFormatter": "esbenp.prettier-vscode"
    },
    "editor.inlineSuggest.enabled": true,
    "github.copilot.enable": {
```

VSC Jest extension

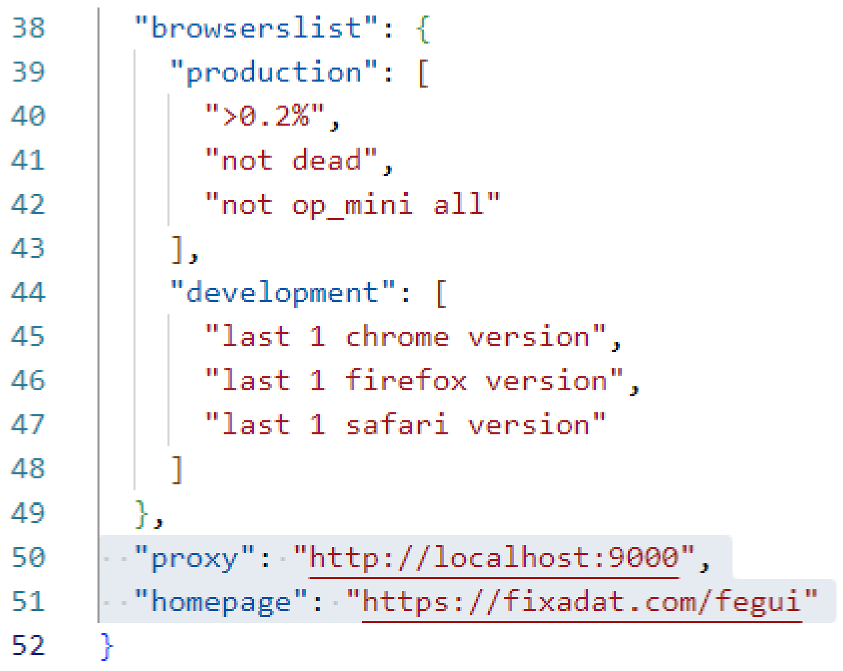
Two Become One 1

If you have not done so yet, read the following pages:

- Proxying in Development[148]
- Building for Relative Paths[149]

Dude, where's my API?

Because the GUI and the API are two different apps during development, API requests need to be proxied. Simply add the key-value pair `"proxy": "http://localhost:9000/"` to `Fixadat/fegui/package.json`.



```
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "proxy": "http://localhost:9000",
  "homepage": "https://fixadat.com/fegui"
}
```

API requests proxy

[148]https://create-react-app.dev/docs/proxying-api-requests-in-development
[149]https://create-react-app.dev/docs/deployment/#building-for-relative-paths

Prepare for Launch

Since we are using React for the GUI (and not Twirl[150]), the generated `HomeController` need not return any result. As a matter of fact, we can delete `Fixadat/beapi/app/controllers/HomeController.scala` as well as `Fixadat/beapi/app/views` and `Fixadat/beapi/test/controllers`. Instead, we need a controller that returns the HTML page with the DOM container for React and the React-generated (from Play's point of view static) files.

To this end, introduce `Fixadat/beapi/app/gui/ReactController.scala`[151]:

```scala
package gui

import play.api.Environment
import play.api.mvc.AnyContent
import play.api.mvc.BaseController
import play.api.mvc.ControllerComponents
import play.api.mvc.Request
import play.filters.csrf.CSRF

import javax.inject.Inject
import javax.inject.Singleton
import scala.io.Codec
import scala.io.Source

@Singleton
class ReactController @Inject() (
    val controllerComponents: ControllerComponents,
    val env: Environment
) extends BaseController {

  val is = env.classLoader.getResourceAsStream("public/build/index.html")
  val bufferedSource = Source.createBufferedSource(
    inputStream = is,
    close = () => is.close()
  )(Codec.UTF8)
  val stringBuilder = bufferedSource.addString(new StringBuilder())
  val indexHtml = stringBuilder.mkString

  def guiFile(reactFile: String) = Action {
    implicit request: Request[AnyContent] =>
      implicit val ec: scala.concurrent.ExecutionContext =
```

[150]https://www.playframework.com/documentation/latest/ScalaTemplates
[151]https://github.com/Squoss/Fixadat/blob/main/beapi/app/gui/ReactController.scala

```
      scala.concurrent.ExecutionContext.global
    Ok.sendResource(
      s"public/build/$reactFile",
      env.classLoader
    ) // TODO/FIXME: check for path-traversal vulnerability
  }

  def guiRoute(reactRoute: String) = Action {
    implicit request: Request[AnyContent] =>
    val token =
      CSRF.getToken // // https://www.playframework.com/documentation/latest/Scala\
Csrf#Getting-the-current-token
    Ok(indexHtml.replace("REPLACE_CSRF_TOKEN", token.get.value))
      .as("text/html")
  }
}
```

In Fixadat/beapi/conf/routes[152], the first mapping needs to be modified accordingly and two mappings need to be added at the very end:

```
GET /                 gui.ReactController.guiRoute(reactRoute = "")
…
GET /fegui/*reactFile gui.ReactController.guiFile(reactFile)
GET /*reactRoute      gui.ReactController.guiRoute(reactRoute)
```

We are going to revisit the new GUI controller as well as the routes mappings in later chapters. For the time being, note how the controller (pre-)loads and returns the exact same HTML page, whether the request path is only `/` or `/legalese/pp` or any other path that does not start with `/fegui/`. The latter prefix is for the React-generated files. Simply add the key-value pair `"homepage": "https://fixadat.com/fegui"` to `Fixadat/fegui/package.json` in order for React to be aware of it.

We can add a simple HTML page (without DOM container) as `index.html` to `Fixadat/beapi/public/build` to serve as a placeholder during development:

[152]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/routes

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fi\
t=no">
    <title>Hello, Fixadat!</title>
    <meta name="csrf-token" content="REPLACE_CSRF_TOKEN">
  </head>
  <body>
    <h1>Hello, API!</h1>
    <p>On <code>localhost</code>, i.e., during development, the GUI is available at \
<a href="http://localhost:3000/">port 3000</a>.</p>
  </body>
</html>
```

As we shall see below, it will be overwritten with React’s `index.html`, which includes the DOM container, when the production app is built.

Security as a Forethought

As you certainly know, you cannot develop and deploy any Web app without considering security. And I mean *you* personally. Your organization also needs to seriously consider data protection and privacy[153] as those are a question of your organization's policy (vis-à-vis its customers, its products' users, etc.), some of which is enforced with security mechanisms.

Considering a Web app's security means considering aspects such as identification & authentication, authorization, software security, security software, and many more. This chapter is not about any of these aspects. This chapter is about hardening the app against some common Web threats that are virtually independent of your app and that services such as Mozilla Observatory[154] (bookmark it!) can partially measure.

If you are new to Web application security, peruse the OWASP[155] resources, in particular the OWASP Top Ten[156] and the OWASP Cheat Sheet Series[157], and read WASEC: Web Application Security for the everyday software engineer[158] as a gentle introduction to the subject.

Hardening a Web app is more of a back-end responsibility, but the front-end is not completely off the hook.

[153]http://williamstallings.com/Privacy/
[154]https://observatory.mozilla.org/
[155]https://owasp.org/
[156]https://owasp.org/www-project-top-ten/
[157]https://cheatsheetseries.owasp.org/
[158]https://leanpub.com/wasec

Hardening the back-end

If you have not done so yet, read the following pages:

- Protecting against Cross Site Request Forgery[159]
- HTTP filters[160]
- Configuring the application secret[161]
- Configuring the session cookie[162]
- Play HTTP filters[163]
- Configuring security headers[164]
- Configuring CORS[165]
- Configuring CSP[166]
- Configuring allowed hosts[167]
- Configuring HTTPS redirect[168]

As we shall see in the configuration chapter, the configuration for a Play app is found in its `application.conf` file (`Fixadat/beapi/conf/application.conf` in our case). Furthermore, Play provides various filters, most of which are security filters. Not all of Play's security mechanisms which allow for hardening a Play app are filters, but all can be configured.

When running Fixadat, you must see the following output (in any order):

```
play.filters.csrf.CSRFFilter
play.filters.headers.SecurityHeadersFilter
play.filters.hosts.AllowedHostsFilter
play.filters.csp.CSPFilter
play.filters.https.RedirectHttpsFilter
```

This is necessary but not sufficient for Play to be configured securely.

The first three filters listed above are enabled by default; to make it explicit, include the key-value pair `play.http.filters = play.api.http.EnabledFilters`. The last two filters listed above must be enabled explicitly by adding the key-value pairs `play.filters.enabled += play.filters.csp.CSPFilter`, and `play.filters.enabled +=`

[159]https://www.playframework.com/documentation/latest/ScalaCsrf
[160]https://www.playframework.com/documentation/latest/ScalaHttpFilters
[161]https://www.playframework.com/documentation/latest/ApplicationSecret
[162]https://www.playframework.com/documentation/latest/SettingsSession
[163]https://www.playframework.com/documentation/latest/Filters
[164]https://www.playframework.com/documentation/latest/SecurityHeaders
[165]https://www.playframework.com/documentation/latest/CorsFilter
[166]https://www.playframework.com/documentation/latest/CspFilter
[167]https://www.playframework.com/documentation/latest/AllowedHostsFilter
[168]https://www.playframework.com/documentation/latest/RedirectHttpsFilter

`play.filters.https.RedirectHttpsFilter`. Note that the CORS filter[169] is not enabled[170]; Fixadat is a self-contained system[171] whose API is not meant for third-party clients (yet).

Testing

If the (default) filters interfere with (unit) tests, refer to

- Testing Default Filters[a]
- Testing CSRF[b]
- Testing with CSRFFilter[c]
- Testing[d]
- Testing with AllowedHostsFilter[e]

[a]https://www.playframework.com/documentation/latest/Filters#Testing-Default-Filters
[b]https://www.playframework.com/documentation/latest/ScalaCsrf#Testing-CSRF
[c]https://www.playframework.com/documentation/latest/Filters#Testing-with-CSRFFilter
[d]https://www.playframework.com/documentation/latest/AllowedHostsFilter#Testing
[e]https://www.playframework.com/documentation/latest/Filters#Testing-with-AllowedHostsFilter

Application Secret

Play requires an application secret, which defaults to `changeme`, which in turn would not be accepted in production as it would be insecure. In production, we are going to set it via an environment variable[172]. We are going to define the environment variable (named `APPLICATION_SECRET`) in the last part. Right now, we only need to add the key-value pair `play.http.secret.key = ${?APPLICATION_SECRET}` so that Play looks for it.

Session Cookie

Unless you have really really good reasons (Do you really?) not to, you should harden all your app's cookies by setting `Secure`[173], `SameSite=Strict`[174], and `HttpOnly`[175]. You can configure Play to do so for the session cookie by adding the key-value pairs `play.http.session.secure = true`, `play.http.session.sameSite = "strict"`, and `play.http.session.httpOnly = true`.

[169]https://www.playframework.com/documentation/latest/CorsFilter
[170]If your app needs to allow for Cross Origin Resource Sharing, heed OWASP's advice.
[171]https://scs-architecture.org/
[172]https://www.playframework.com/documentation/latest/ApplicationSecret#Environment-variables
[173]https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#secure-attribute
[174]https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#samesite-attribute
[175]https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#httponly-attribute

Cross-Site Request Forgery (CSRF)

In order to prevent[176] CSRF attacks[177], a CSRF token must be included with certain HTTP requests. Remember that *"By default, Play will require a CSRF check when* ***all*** *of the following* ***are true****:"*

- The request method is not GET, HEAD or OPTIONS.
- The request has one or more Cookie or Authorization headers.
- The CORS filter is not configured to trust the request's origin.

The first requirement implies that an API better be RESTful[178]; more specifically, GET, HEAD, and OPTIONS requests must not have any side effects. The second requirement can be re-configured to protect all requests: add the key-value pair `play.filters.csrf.header.protectHeaders = null`. The third requirement can be re-configured to NOT trust CORS requests[179]: add the key-value pair `play.filters.csrf.bypassCorsTrustedOrigins = false`.

Since *"CSRF tokens should not be transmitted using cookies"*[180], we are going to use a custom request header[181]. In production, the back-end is going to store the CSRF token in the DOM[182]. Therefore, the front-end must provide a placeholder in `Fixadat/fegui/public/index.html` and set the custom header[183] when it makes certain API calls. In order to test the replacement of the placeholder during development, add the line `<meta name="csrf-token" content="REPLACE_CSRF_TOKEN" />` to `Fixadat/beapi/public/index.html`'s `<head>` section. In order to actually replace the placeholder, overwrite the implementation of the `index()` method in `Fixadat/beapi/app/controllers/HomeController.scala` with the following one (and add `import play.filters.csrf.CSRF`):

```
  def index() = Action { implicit request: Request[AnyContent] =>
    val token =
      CSRF.getToken // // https://www.playframework.com/documentation/latest/ScalaCs\
rf#Getting-the-current-token
    Ok(string.replace("REPLACE_CSRF_TOKEN", token.get.value))
      .as("text/html")
  }
```

In production, Fixadat's front-end and back-end will agree on what the value of the CSRF token is. During development, they will not. Instead, the back-end can be configured to skip the CSRF check[184]

[176]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
[177]https://owasp.org/www-community/attacks/csrf
[178]https://dpunkt.de/produkt/rest-und-http-2/
[179]https://www.playframework.com/documentation/latest/ScalaCsrf#Trusting-CORS-requests
[180]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-pattern
[181]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#use-of-custom-request-headers
[182]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#storing-the-csrf-token-value-in-the-dom
[183]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#overriding-defaults-to-set-custom-header
[184]https://www.playframework.com/documentation/latest/ScalaCsrf#Plays-CSRF-protection

when the value of the `Csrf-Token` header is `REPLACE_CSRF_TOKEN`. For obvious reasons, we do not want to add the key-value pair `play.filters.csrf.header.bypassHeaders.Csrf-Token = "REPLACE_CSRF_TOKEN"` to `application.conf`. Instead, add the following configuration as `insecureLocalhost.conf` (the inverse of a production configuration file[185], so to speak) to `Fixadat/beapi/conf` and start the back-end locally with `[Fixadat] $ run -Dconfig.file=conf/insecureLocalhost.conf` from now on:

```
include "application"

play.filters.csrf.header.bypassHeaders.Csrf-Token = "REPLACE_CSRF_TOKEN"
```

Security Headers

Play supports various headers[186] to enhance security[187]. In two cases, we can be even stricter than Play's defaults[188] by adding the key-value pairs `play.filters.headers.permittedCrossDomainPolicies = "none"` and `play.filters.headers.referrerPolicy = "strict-origin-when-cross-origin"`. Futhermore, let us make it explicit that we do not allow for action-specific overrides[189] by adding the key-value pair `play.filters.headers.allowActionSpecificHeaders = false`.

Content Security Policy (CSP)

Play features a dedicated CSP[190] filter. As we want to have a tight basic CSP policy[191] (namely, `Content-Security-Policy: default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src 'self';`), add the following key-value pairs:

```
play.filters.csp.directives.default-src = "'none'"
play.filters.csp.directives.connect-src = "'self'"
play.filters.csp.directives.font-src = "'self'"
play.filters.csp.directives.img-src = "'self'"
play.filters.csp.directives.manifest-src = "'self'"
play.filters.csp.directives.script-src = "'self'"
play.filters.csp.directives.style-src = "'self'"
```

Furthermore, we want mixed content to be blocked[192] by adding the key-value pair `play.filters.csp.directives.block-all-mixed-content = ""` and click-jacking to be prevented[193] by adding the key-value pair `play.filters.csp.directives.frame-ancestors = "'none'"`.

[185] https://www.playframework.com/documentation/latest/ApplicationSecret#Production-configuration-file
[186] https://owasp.org/www-project-secure-headers/
[187] https://cheatsheetseries.owasp.org/cheatsheets/HTML5_Security_Cheat_Sheet.html#http-headers-to-enhance-security
[188] https://www.playframework.com/documentation/latest/SecurityHeaders#Configuring-the-security-headers
[189] https://www.playframework.com/documentation/latest/SecurityHeaders#Action-specific-overrides
[190] https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
[191] https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html#basic-csp-policy
[192] https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html#mixed-content-policy
[193] https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html#preventing-clickjacking

Allowed Hosts

As you know[194], Play allows for limiting the hosts that can make requests by allow-listing those. We would like fixadat.com (including any and all sub-domains) and localhost to be allow-listed, which is why we add the key-value par `play.filters.hosts.allowed = [".fixadat.com", "localhost", ${?PAAS_DOMAIN}]`. The last entry in the array value allows for substituting the domain of the PaaS provider via an environment variable.

In the last part, we are going to allow-list `"fixadat.cleverapps.io"`. If we would like to allow-list further domains of our PaaS provider Clever Cloud, fixadat-test.cleverapps.io for instance, we have to allow-list them explicitly and must not allow-list cleverapps.io in general!

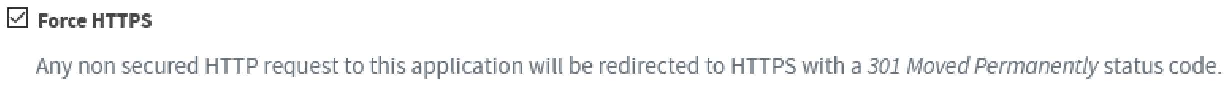
Redirecting HTTP to HTTPS

In production, enforcing HTTP over TLS or simply HTTPS is a must. As you know[195], Fixadat further instructs browsers to switch to HTTPS even if the user entered only "http://" by taking advantage of Strict Transport Security[196]: `play.filters.https.strictTransportSecurity = "max-age=31536000; includeSubDomains"`

If you would like to already enforce HTTPS during development[197], add the key-value pair `play.filters.https.redirectEnabled = true`. If you would like to, you could configure a custom TLS certificate for the Play project[198] or even for your browser[199], but the latter would be akin to playing with fire, so beware!

Note that in production, determining whether a request was sent over TLS requires configuring trusted proxies[200], which we are going to do in the last part. That's because the TLS connection terminates at the edge and is not handled by Play[201] itself.

In the last part, we are also configuring our PaaS provider Clever Cloud to enforce HTTPS.



☑ **Force HTTPS**

Any non secured HTTP request to this application will be redirected to HTTPS with a *301 Moved Permanently* status code.

Force HTTPS

[194]from https://www.playframework.com/documentation/latest/ScalaContentNegotiation#Language
[195]from https://www.playframework.com/documentation/latest/RedirectHttpsFilter
[196]https://www.playframework.com/documentation/latest/RedirectHttpsFilter#Strict-Transport-Security
[197]https://www.playframework.com/documentation/latest/RedirectHttpsFilter#Enabling-the-HTTPS-filter
[198](https://www.playframework.com/documentation/latest/ConfiguringHttps#SSL-Certificates-from-a-keystore)
[199]https://github.com/FiloSottile/mkcert
[200]https://www.playframework.com/documentation/latest/HTTPServer#Configuring-trusted-proxies
[201]https://www.playframework.com/documentation/latest/ConfiguringHttps#Production-usage-of-HTTPS

Hardening the front-end

Content Security Policy (CSP)

Even though the content security policy[202] (CSP) is configured at and served by the back-end, there is something we have to do for it (or rather because of it) in the front-end. Since we want a CSP[203] that does **not** allow `'unsafe-inline'`[204] (let alone `'unsafe-eval'`), React needs to be configured not to generate any inline scripts. As you know[205], you need to add a file called `.env` to `Fixadat/fegui` and add the line `INLINE_RUNTIME_CHUNK=false` to it.

Cross-Site Request Forgery (CSRF)

In order to prevent[206] CSRF attacks[207], a CSRF token must be included with certain HTTP requests. Since *"CSRF tokens should not be transmitted using cookies"*[208], we are going to use a custom request header[209]. In production, the back-end is going to store the CSRF token in the DOM[210]. Therefore, the front-end must provide a placeholder in `Fixadat/fegui/public/index.html` and set the custom header[211] when it makes certain API calls. For the former, add the line `<meta name="csrf-token" content="REPLACE_CSRF_TOKEN" />` to `Fixadat/fegui/public/index.html`'s `<head>` section. For the latter, add the following helper functions as `fetchJson.ts` (which is a mash-up between https://www.carlrippon.com/fetch-with-async-await-and-typescript/ and https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#xmlhttprequest-native-javascript) to `Fixadat/fegui/src`:

[202]https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy
[203]https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
[204]https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/script-src
[205]from https://create-react-app.dev/docs/production-build/, https://create-react-app.dev/docs/adding-custom-environment-variables/, and https://create-react-app.dev/docs/advanced-configuration/
[206]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html
[207]https://owasp.org/www-community/attacks/csrf
[208]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#synchronizer-token-pattern
[209]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#use-of-custom-request-headers
[210]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#storing-the-csrf-token-value-in-the-dom
[211]https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html#overriding-defaults-to-set-custom-header

```
// https://www.carlrippon.com/fetch-with-async-await-and-typescript/

interface HttpResponse<T> extends Response {
  parsedBody?: T;
}
async function fetchJson<T>(request: Request): Promise<HttpResponse<T>> {
  // https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Preve\
ntion_Cheat_Sheet.html#xmlhttprequest-native-javascript
  if (!/^(GET|HEAD|OPTIONS)$/.test(request.method)) {
    const csrf_token = document.querySelector("meta[name='csrf-token']")!.getAttribu\
te("content");
    request.headers.append("Csrf-Token", csrf_token!);
  }

  const response: HttpResponse<T> = await fetch(request);
  try {
    response.parsedBody = await response.json();
  } catch (ex) { }

  if (!response.ok) {
    throw new Error(response.statusText);
  }
  return response;
}

export async function get<T>(path: string, accessToken: string, args: RequestInit = \
{ method: "get", mode: "same-origin", credentials: "same-origin", cache: "no-store",\
 redirect: "error", headers: { "X-Access-Token": accessToken } }): Promise<HttpRespo\
nse<T>> {
  return await fetchJson<T>(new Request(path, args));
}

export async function patch<T>(
  path: string,
  accessToken: string,
  body: any,
  args: RequestInit = { method: "PATCH", body: JSON.stringify(body), mode: "same-ori\
gin", credentials: "same-origin", cache: "no-store", redirect: "error", headers: { "\
Content-Type": "application/json", "X-Access-Token": accessToken } },
): Promise<HttpResponse<T>> {
  return await fetchJson<T>(new Request(path, args));
}
```

```
export async function post<T>(
  path: string,
  accessToken = "",
  body = {},
  args: RequestInit = { method: "POST", body: JSON.stringify(body), mode: "same-orig\
in", credentials: "same-origin", cache: "no-store", redirect: "error", headers: { "C\
ontent-Type": "application/json", "X-Access-Token": accessToken } },
): Promise<HttpResponse<T>> {
  return await fetchJson<T>(new Request(path, args));
}

export async function put<T>(
  path: string,
  accessToken: string,
  body: any,
  args: RequestInit = { method: "PUT", body: JSON.stringify(body), mode: "same-origi\
n", credentials: "same-origin", cache: "no-store", redirect: "error", headers: { "Co\
ntent-Type": "application/json", "X-Access-Token": accessToken } },
): Promise<HttpResponse<T>> {
  return await fetchJson<T>(new Request(path, args));
}
```

Cross-Site Scripting (XSS)

In order to prevent[212] Stored or Reflected XSS attacks[213] and to prevent[214] DOM-based attacks[215], the GUI's content must be properly escaped. Luckily, React takes care of escaping[216]. If you really really have to circumvent React escaping[217] some content (Do you really?), do yourself a favor and at least avoid any user-generated content as well as content consumed from third parties (e.g., via their APIs)!

What about TLS?

In production, enforcing HTTP over TLS or simply HTTPS is a must. If you already enable or even enforce HTTPS in the back-end during development, you may want to do so in the front-end as well. Simply add the line `HTTPS=true` to `Fixadat/fegui/.env`. If you would like to, you could configure a custom TLS certificate for your front-end project[a] or even for your browser[b], but the latter would

[212]https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
[213]https://owasp.org/www-community/attacks/xss/
[214]https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html
[215]https://owasp.org/www-community/attacks/DOM_Based_XSS
[216]https://reactjs.org/docs/introducing-jsx.html#jsx-prevents-injection-attacks
[217]https://reactjs.org/docs/dom-elements.html#dangerouslysetinnerhtml

be akin to playing with fire, so beware!

[a](https://create-react-app.dev/docs/using-https-in-development/#custom-ssl-certificate)
[b]https://github.com/FiloSottile/mkcert

Rinse & Repeat

For as long as your project is deployed, you will not be done with security; you need to strive for continuous security[218]. With respect to this chapter, you need to answer at least two questions on a regular basis (e.g., at the end of a Sprint[219] before you deliver the Increment[220] if you happen to employ Scrum with Essence[221] (or without)):

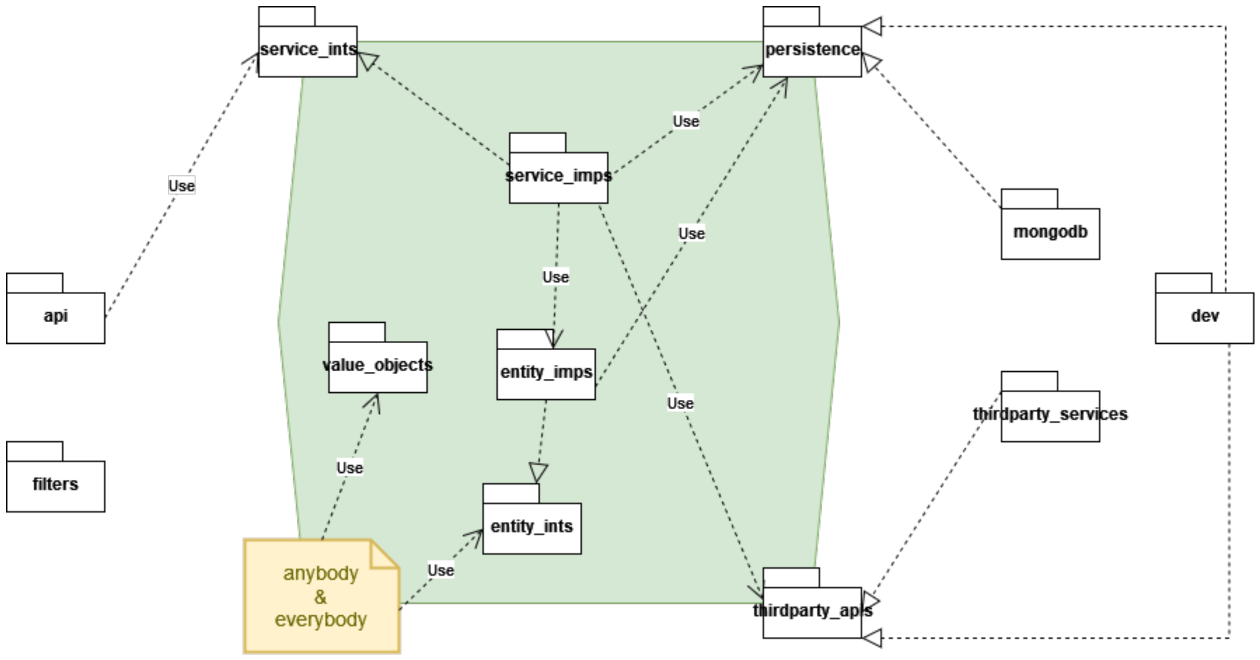
1. Do I need to update Play[222] or another dependency[223] because of a newly discovered security vulnerability?
2. Do I need to update React[224] or another package[225] because of a newly discovered security vulnerability?
3. Are the steps taken above still necessary[226] and sufficient?

As for 1 and 2, you could automate some parts within your repository[227] and/or pipeline[228] and/or …

[218]https://www.manning.com/books/securing-devops
[219]https://www.scrumguides.org/scrum-guide.html#the-sprint
[220]https://www.scrumguides.org/scrum-guide.html#increment
[221]https://www.scruminc.com/better-scrum-with-essence/
[222]https://www.playframework.com/documentation/latest/Migration28
[223]https://github.com/albuch/sbt-dependency-check
[224]https://create-react-app.dev/docs/updating-to-new-releases
[225]https://docs.npmjs.com/cli/v6/commands/npm-audit
[226]https://infosec.mozilla.org/guidelines/web_security
[227]https://docs.github.com/en/free-pro-team@latest/github/managing-security-vulnerabilities
[228]https://support.snyk.io/hc/en-us/sections/360001152577-CI-CD-integrations

So Long, MVC!

The Play project was obviously generated witht the Model-View-Controller pattern in mind. Since I prefer a clean[229]/hexagonal[230]/onion[231] architecture[232], however, we are going to restructure the project accordingly. (The advantages and disadvantages of moving from Model-View-Controller (MVC) to Ports & Adapters are very well covered in *Get Your Hands Dirty on Clean Architecture*[233].) The goal is to enforce the architecture depicted below, where the green hexagon and its border represent the domain (quite in the sense of Domain-Driven Design[234]).



Packages

The Controllers are Dead, Long Live the Controllers!

In the last chapter, we just got rid of the Play views. In order to emphasize that the controllers are first & foremost providing an API, we rename `Fixadat/beapi/app/controllers` to `Fixadat/beapi/app/api` and change `HomeController`'s package also from `controllers` to `api`, both in `Fixadat/beapi/app/api/HomeController.scala` and `Fixadat/beapi/conf/routes`.

[229] https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
[230] https://alistair.cockburn.us/hexagonal-architecture/
[231] https://jeffreypalermo.com/tag/onion-architecture/
[232] See also https://youtu.be/UakcxHQ1uDg and *Domain Modeling Made Functional*, a book worth reading.
[233] https://leanpub.com/get-your-hands-dirty-on-clean-architecture
[234] https://leanpub.com/ddd-by-example

sbt Subproject

As you know[235], a Play project can have a simple library subproject. By organizing the domain as such a subproject (`Fixadat/beapi/reinraum`), the build system can enforce its independence from the “harsh world” (Web, DBMS, etc.) around it. To create the subproject for the domain, add folder `reinraum` in `Fixadat/beapi/` and replace the line `lazy val root = (project in file(".")).enablePlugins(PlayScala)` in `Fixadat/beapi/build.sbt` by the following lines:

```
lazy val root = (project in file("."))
    .enablePlugins(PlayScala)
    .aggregate(reinraum)
    .dependsOn(reinraum)
lazy val reinraum = project
```

Then add folders `src` to `Fixadat/beapi/reinraum`, `main` as well as `test` to `Fixadat/beapi/reinraum/src`, `scala` to `Fixadat/beapi/reinraum/src/main` as well as to `Fixadat/beapi/reinraum/src/test`, and the following configuration as `build.sbt` to `Fixadat/beapi/reinraum`:

```
scalaVersion := "2.13.10"

libraryDependencies += guice
libraryDependencies += "com.google.inject" % "guice" % "5.1.0" // bumping the Guice \
version manually allows for using Play 2.8 with Java 17

libraryDependencies += "org.scalatest" %% "scalatest" % "3.2.7" % Test
```

Note how the domain’s `build.sbt` includes the regular `scalatest` while the Play project’s includes `scalatestplus-play`.

ArchUnit tests

There is still substantial risk for quick & dirty shortcuts. We introduce ArchUnit[236] test suites that enforce our dependency rules in order to ensure that the dependencies around, on, and within the domain do not become a tangled mess.

In both `Fixadat/beapi/build.sbt` and `Fixadat/beapi/reinraum/build.sbt`, add the line `libraryDependencies += "com.tngtech.archunit" % "archunit" % "0.21.0" % Test`. In `Fixadat/beapi/test`, add the following code as `DependencyRulesTestSuite.scala`:

[235]from https://www.playframework.com/documentation/latest/sbtSubProjects#Adding-a-simple-library-sub-project
[236]https://www.archunit.org/

```scala
import com.tngtech.archunit.core.importer.ClassFileImporter
import com.tngtech.archunit.lang.syntax.ArchRuleDefinition.noClasses
import org.scalatest.funsuite.AnyFunSuite

class DependencyRulesTestSuite extends AnyFunSuite {

  val AKKA = "akka.."
  val API = "api.."
  val DEFAULT = ""
  val DEV = "dev.."
  val DOMAIN_ENTITYINTS = "domain.entity_interfaces.."
  val DOMAIN_SERVICEINTS = "domain.service_interfaces.."
  val DOMAIN_PERSISTENCE = "domain.persistence.."
  val DOMAIN_VALUEOBJECTS = "domain.value_objects.."
  val FILTERS = "filters.."
  val JAVA = "java.."
  val JAVAX = "javax.."
  val MAILJET = "com.mailjet.."
  val MONGODB_ADAPTER =
    "mongodb.." // the MongoDB driver starts its packages with com or org
  val MONGODB_DRIVER = Seq("org.bson..", "org.mongodb..")
  val PLAY_API = "play.api.." // "play.." would also include Play's Java API
  val PLAY_CORE = "play.core.."
  val PLAY_FILTERS = "play.filters.."
  val PHONENUMBERS = "com.google.i18n.phonenumbers.."
  val ROUTER = "router.."
  val SCALA = "scala.."
  val THIRDPARTY_APIS = "thirdparty_apis.."
  val THIRDPARTY_SERVICES = "thirdparty_services.."
  val VALIDATORS = "org.apache.commons.validator.."

  val NOT_THE_APP =
    Seq(
      JAVA,
      JAVAX,
      SCALA,
      AKKA,
      PLAY_API,
      PLAY_CORE,
      PLAY_FILTERS,
      ROUTER,
      PHONENUMBERS,
      VALIDATORS
    )
  val THE_APP_OUTSIDE_OF_THE_DOMAIN =
    Seq(DEFAULT, API, DEV, FILTERS, MONGODB_ADAPTER, THIRDPARTY_SERVICES)

  val classes =
    new ClassFileImporter().importPackages(THE_APP_OUTSIDE_OF_THE_DOMAIN: _*)

  test(
    "the controllers depend on themselves, the SPI, the (abstract) types, and the va\
lue objects only"
  ) {

    noClasses()
      .that()
      .resideInAPackage(API)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ API :+ DOMAIN_SERVICEINTS :+ DOMAIN_ENTITYINTS :+ DOMAIN_VAL\
UEOBJECTS): _*
      )
      .check(classes)
  }

  test("nothing depends on the controllers") {

    noClasses()
      .that()
      .resideOutsideOfPackages(API, ROUTER)
      .should()
      .dependOnClassesThat()
      .resideInAPackage(API)
      .check(classes)
  }

  // we don't care what DEV implementations depend on

  test("nothing depends on DEV implementations") {

    noClasses()
      .that()
      .resideOutsideOfPackage(DEV)
      .and()
      .doNotHaveSimpleName("Module")
      .should()
      .dependOnClassesThat()
      .resideInAPackage(DEV)
      .check(classes)
  }

  test("the filters depend on themselves only") {

    noClasses()
      .that()
      .resideInAPackage(FILTERS)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ "play.mvc.."): _*
      ) // "play.mvc.." covers play.mvc.EssentialFilter
      .check(classes)
  }

  test("nothing depends on the filters") {

    noClasses()
      .that()
      .resideOutsideOfPackage(FILTERS)
      .should()
      .dependOnClassesThat()
      .resideInAPackage(FILTERS)
      .check(classes)
  }

  test(
    "the MongoDB adapter depends on itself and the repositories only"
  ) {

    noClasses()
      .that()
      .resideInAPackage(MONGODB_ADAPTER)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP ++ MONGODB_DRIVER :+ MONGODB_ADAPTER :+ DOMAIN_ENTITYINTS :+ DO\
MAIN_PERSISTENCE): _*
```

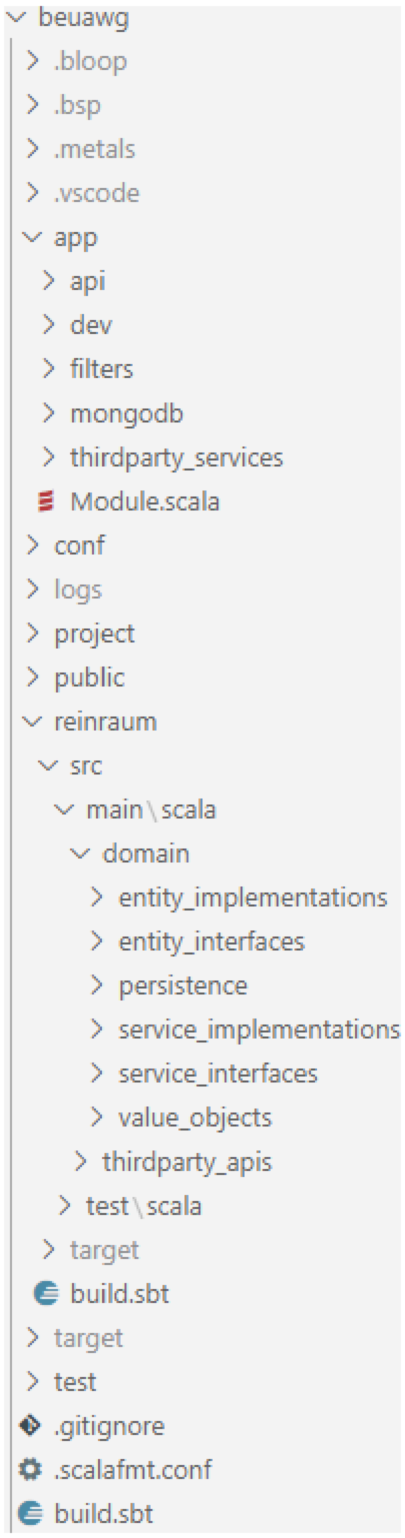
```
      )
      .check(classes)
  }

  test("nothing depends on the MongoDB adapter") {

    noClasses()
      .that()
      .resideOutsideOfPackage(MONGODB_ADAPTER)
      .and()
      .doNotHaveSimpleName("Module")
      .should()
      .dependOnClassesThat()
      .resideInAPackage(MONGODB_ADAPTER)
      .check(classes)
  }

  test(
    "third-party services depend on themselves and the third-party APIs only"
  ) {

    noClasses()
      .that()
      .resideInAPackage(THIRDPARTY_SERVICES)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ DOMAIN_VALUEOBJECTS :+ MAILJET :+ THIRDPARTY_APIS :+ THIRDPA\
RTY_SERVICES): _*
      )
      .check(classes)
  }

  test("nothing depends on third-party services") {

    noClasses()
      .that()
      .resideOutsideOfPackage(THIRDPARTY_SERVICES)
      .and()
      .doNotHaveSimpleName("Module")
      .should()
      .dependOnClassesThat()
      .resideInAPackage(THIRDPARTY_SERVICES)
      .check(classes)
  }
}
```

In `Fixadat/beapi/reinraum/src/test/scala`, add the following code as `DependencyRulesTestSuite.scala`:

```
import com.tngtech.archunit.core.importer.ClassFileImporter
import com.tngtech.archunit.lang.syntax.ArchRuleDefinition.noClasses
import org.scalatest.funsuite.AnyFunSuite

class DependencyRulesTestSuite extends AnyFunSuite {

  val DEFAULT = ""
  val DOMAIN_ENTITYIMPS = "domain.entity_implementations.."
  val DOMAIN_ENTITYINTS = "domain.entity_interfaces.."
  val DOMAIN_PERSISTENCE = "domain.persistence.."
  val DOMAIN_SERVICEIMPS = "domain.service_implementations.."
  val DOMAIN_SERVICEINTS = "domain.service_interfaces.."
  val DOMAIN_VALUEOBJECTS = "domain.value_objects.."
  val JAVA = "java.."
  val JAVAX = "javax.."
  val PHONENUMBERS = "com.google.i18n.phonenumbers.."
  val SCALA = "scala.."
  val THIRDPARTY_APIS = "thirdparty_apis.."

  val NOT_THE_APP =
    Seq(JAVA, JAVAX, SCALA, PHONENUMBERS)
  val THE_APP_INSIDE_THE_DOMAIN =
    Seq(
      DOMAIN_ENTITYIMPS,
      DOMAIN_ENTITYINTS,
      DOMAIN_PERSISTENCE,
      DOMAIN_SERVICEIMPS,
      DOMAIN_SERVICEINTS,
      DOMAIN_VALUEOBJECTS
    )

  val classes =
    new ClassFileImporter().importPackages(
      (THE_APP_INSIDE_THE_DOMAIN :+ THIRDPARTY_APIS): _*
    )

  test("the domain depends on itself and third-party APIs only") {

    noClasses()
      .that()
      .resideInAnyPackage(THE_APP_INSIDE_THE_DOMAIN: _*)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP ++ THE_APP_INSIDE_THE_DOMAIN :+ THIRDPARTY_APIS): _*
      )
      .check(classes)
  }

  test(
    "the repositories port depends on itself and the domain types only"
  ) {

    noClasses()
      .that()
      .resideInAPackage(DOMAIN_PERSISTENCE)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ DOMAIN_PERSISTENCE :+ DOMAIN_ENTITYINTS :+ DOMAIN_VALUEOBJEC\
TS): _*
      )
      .check(classes)
  }

  test(
    "besides the repositories port, only the domain services and domain entities dep\
end on the repositories port"
  ) {

    noClasses()
      .that()
      .resideOutsideOfPackages(
        DOMAIN_PERSISTENCE,
        DOMAIN_SERVICEIMPS,
        DOMAIN_ENTITYIMPS
      )
      .should()
      .dependOnClassesThat()
      .resideInAPackage(DOMAIN_PERSISTENCE)
      .check(classes)
  }

  test("the services port depends on itself and the domain types only") {

    noClasses()
      .that()
      .resideInAPackage(DOMAIN_SERVICEINTS)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ DOMAIN_SERVICEINTS :+ DOMAIN_ENTITYINTS :+ DOMAIN_VALUEOBJEC\
TS): _*
      )
      .check(classes)
  }

  test("only the domain services depend on the services port") {

    noClasses()
      .that()
      .resideOutsideOfPackages(DOMAIN_SERVICEIMPS)
      .should()
      .dependOnClassesThat()
      .resideInAPackage(DOMAIN_SERVICEINTS)
      .check(classes)
  }

  test("the third-party APIs port depends on itself only") {

    noClasses()
      .that()
      .resideInAPackage(THIRDPARTY_APIS)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages((NOT_THE_APP :+ THIRDPARTY_APIS): _*)
      .check(classes)
  }

  test("only the domain services depend on the third-party APIs port") {

    noClasses()
      .that()
```

```
   .resideOutsideOfPackages(DOMAIN_SERVICEIMPS)
   .should()
   .dependOnClassesThat()
   .resideInAPackage(THIRDPARTY_APIS)
   .check(classes)
 }

 test(
  "the domain services depend on themselves, the domain entities, the domain types\
, and the ports only"
 ) {

  noClasses()
   .that()
   .resideInAPackage(DOMAIN_SERVICEIMPS)
   .should()
   .dependOnClassesThat()
   .resideOutsideOfPackages(
    (NOT_THE_APP :+ DOMAIN_ENTITYIMPS :+ DOMAIN_ENTITYINTS :+ DOMAIN_VALUEOBJECT\
S :+ DOMAIN_PERSISTENCE :+ DOMAIN_SERVICEINTS :+ THIRDPARTY_APIS): _*
   )
   .check(classes)
 }

 test("nothing depends on the domain services") {

  noClasses()
   .that()
   .resideOutsideOfPackage(DOMAIN_SERVICEIMPS)
   .should()
   .dependOnClassesThat()
   .resideInAPackage(DOMAIN_SERVICEIMPS)
   .check(classes)
 }

 test(
  "the entities depend on the domain types and the domain (persistence) events onl\
y"
 ) {

  noClasses()
   .that()
   .resideInAPackage(DOMAIN_ENTITYIMPS)
```

```
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ DOMAIN_ENTITYIMPS :+ DOMAIN_ENTITYINTS :+ DOMAIN_VALUEOBJECT\
S :+ DOMAIN_PERSISTENCE): _*
      )
      .check(classes)
  }

  test("the domain types depend on the value objects only") {

    noClasses()
      .that()
      .resideInAPackage(DOMAIN_ENTITYINTS)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ DOMAIN_ENTITYINTS :+ DOMAIN_VALUEOBJECTS): _*
      )
      .check(classes)
  }

  test("the value objects depend on themselves only") {

    noClasses()
      .that()
      .resideInAPackage(DOMAIN_VALUEOBJECTS)
      .should()
      .dependOnClassesThat()
      .resideOutsideOfPackages(
        (NOT_THE_APP :+ DOMAIN_VALUEOBJECTS): _*
      )
      .check(classes)
  }
}
```



```
beuawg
  .bloop
  .bsp
  .metals
  .vscode
  app
    api
    dev
    filters
    mongodb
    thirdparty_services
    Module.scala
  conf
  logs
  project
  public
  reinraum
    src
      main\scala
        domain
          entity_implementations
          entity_interfaces
          persistence
          service_implementations
          service_interfaces
          value_objects
        thirdparty_apis
      test\scala
    target
    build.sbt
  target
  test
  .gitignore
  .scalafmt.conf
  build.sbt
```

Play Project Structure

With your shell, execute the command `sbt test` in `Fixadat/beapi` to test the architecture.

FIXME/TODO

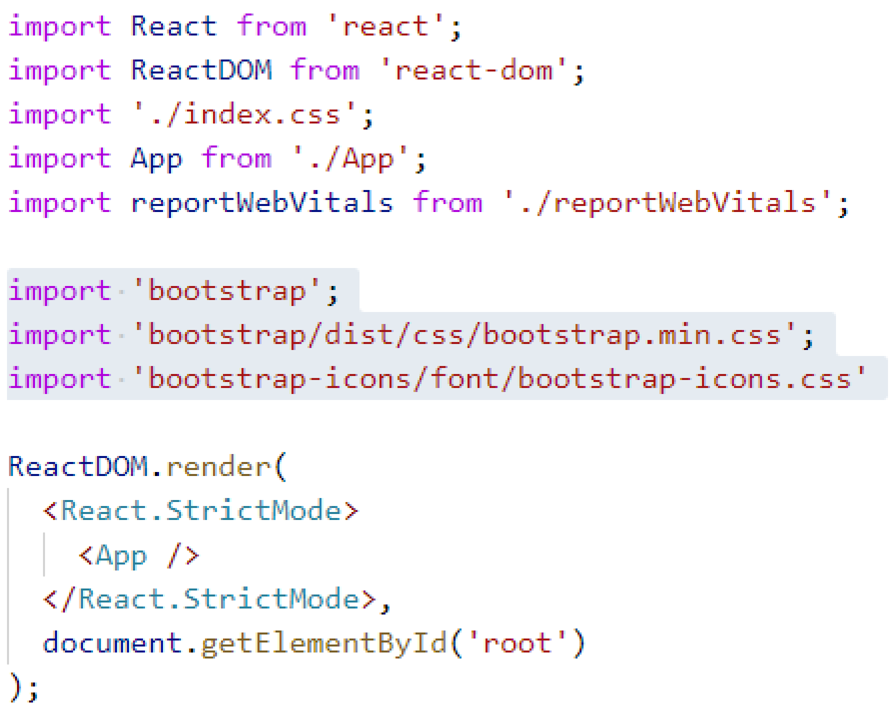
Finally, the Guice[237] module in `Fixadat/beapi/app/Module.scala` takes care of wiring the objects at runtime. We are going to revisit dependency injection in more detail in the remainder of this book.

[237] https://github.com/google/guice

Welcome, User-Friendliness!

Bootstrap

We would like the app's GUI to be responsive and look nice, which is why we are going to add Bootstrap. As you know[238], with your shell, you need to execute the commands `npm install bootstrap` and `npm install @types/bootstrap` as well as `npm install bootstrap-icons` in `Fixadat/fegui`. Note that `Fixadat/fegui/package.json` now has entries for all three imports. Now we can import Bootstrap's CSS, icon font, and logic in `Fixadat/fegui/index.tsx`, our app's entry point. (If you would like to, you could also add a custom theme[239] now or later.)



```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

import 'bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap-icons/font/bootstrap-icons.css'

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Bootstrap imports

[238]from https://create-react-app.dev/docs/adding-bootstrap as well as https://getbootstrap.com/docs/5.1/getting-started/download/#npm, https://getbootstrap.com/docs/5.1/getting-started/webpack/#importing-javascript, https://getbootstrap.com/docs/5.1/getting-started/webpack/#importing-compiled-css, and https://icons.getbootstrap.com/#install

[239]https://www.npmjs.com/package/bootswatch#react-users--create-react-app--or-similar-bundler

React Router

We would like the app's URIs to allow for deep linking[240], which is why we are going to add React Router. As you know[241], with your shell, you need to execute the commands `npm install react-router-dom` and `npm install @types/react-router-dom` in `Fixadat/fegui`. Note that `Fixadat/fegui/package.json` now has entries for both imports.

React Router requires a, well, Router[242] at the top of the component hierarchy which uses its features. We are using a BrowserRouter[243] by adding `import { BrowserRouter } from 'react-router-dom';` to `Fixadat/fegui/src/index.tsx`'s import section and by wrapping `<App />` in `<BrowserRouter />`.

`Fixadat/fegui/src/index.tsx` now (i.e, after having added a custom theme[244] and reorganized the imports) looks as follows:

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter } from 'react-router-dom';
import App from './App';

import 'bootswatch/dist/quartz/bootstrap.min.css';
import 'bootstrap-icons/font/bootstrap-icons.css';
import './index.css';


ReactDOM.render(
  <React.StrictMode>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </React.StrictMode>,
  document.getElementById('root')
);
```

[240] https://en.wikipedia.org/wiki/Deep_linking
[241] from https://create-react-app.dev/docs/adding-a-router/ as well as https://reactrouter.com/web/guides/quick-start/installation
[242] https://reactrouter.com/core/api/Router
[243] https://reactrouter.com/web/api/BrowserRouter
[244] https://bootswatch.com/sketchy/

Business Logic

Interaction Design

As already mentioned in the preface, this is a book about **building the thing right**. Therefore, we simply postulate the existence of a product manager who has figured out the **building the right thing** part with the help of an interaction designer.

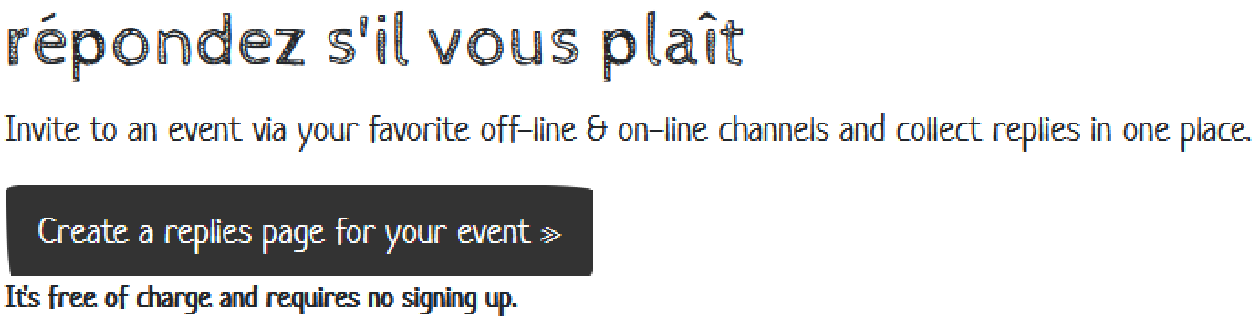
Our product manager and interaction designer have come up with user groups, their interactions, and some (mostly app-independent) boilerplate. (Normally, they would sketch their ideas with a wireframe tool, but I will simply include screenshots of the real thing themed with Bootswatch's Sketchy[245].)

Users and Interactions

There are two kinds of users, hosts and guests.

Hosts

A user clicking a prominent "create" or "new" button on the app's abode must be redirected to a newly created event page.


répondez s'il vous plaît

Invite to an event via your favorite off-line & on-line channels and collect replies in one place.

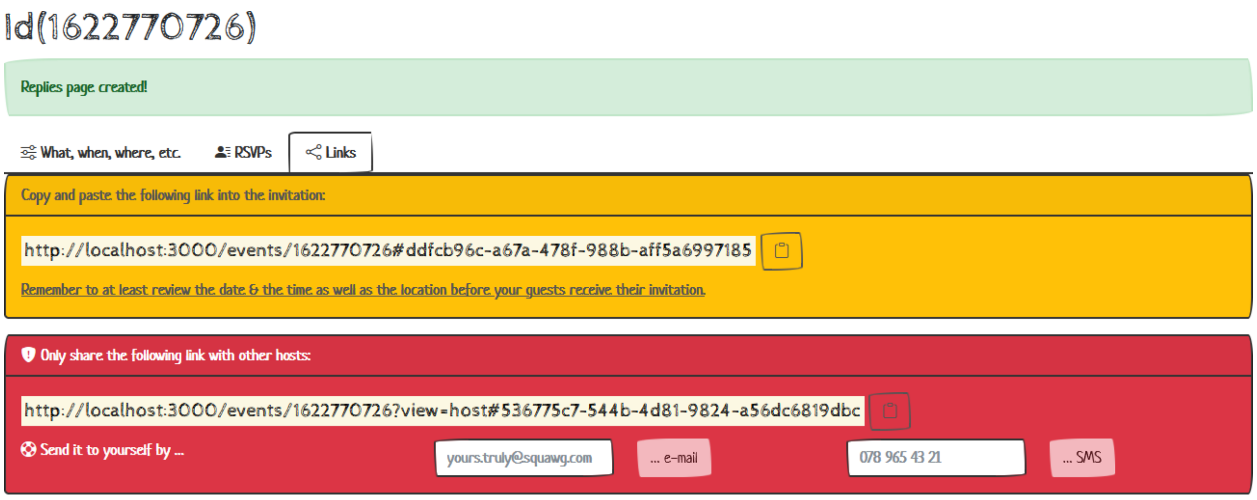
It's free of charge and requires no signing up.

Create Event Page

On the newly created event page, the user is a host, has thus full access rights, and sees three different tabs: Links, Settings, and RSVPs. The Links tab displays two links, the link for guests and the link for hosts and allows for copying and sending the latter (typically to oneself) by e-mail or SMS. The Settings tab allows for editing the event, that is, changing its name, adding a description, fixing its

[245]https://bootswatch.com/sketchy/

date & time, etc. as well as defining whether invitations can be accepted for +1s, too. The RSVPs tab will eventually list all the accepted or declined invitations.


Id(1622770726)

Replies page created!

Copy and paste the following link into the invitation:

http://localhost:3000/events/1622770726#ddfcb96c-a67a-478f-988b-aff5a6997185

Remember to at least review the date & the time as well as the location before your guests receive their invitation.

Only share the following link with other hosts:

http://localhost:3000/events/1622770726?view=host#536775c7-544b-4d81-9824-a56dc6819dbc

Send it to yourself by ...

yours.truly@squawg.com

078 965 43 21

Created Event Page

Guests

Users invited to an event page are guests and can RSVP by entering their name as well as their cell-phone number and indicate a +1.

Boilerplate

The boilerplate consists of one page for acknowledgements (one page for the prices?! FIXE/TODO) and three pages for legal texts. Futhermore, when a user enters a path that is undefined, the app is supposed to respond with a basic Not Found page.

In summary, these are the paths and pages from a user's perspective:

Path	Page
/	the app's abode
/events/EVENT_ID#GUEST_TOKEN	the RSVP page of event EVENT_ID
/events/EVENT_ID#HOST_TOKEN	redirected to /event/EVENT_-ID/RSVPs#HOST_TOKEN
/events/EVENT_ID/links#HOST_-TOKEN	the Links tab of event EVENT_ID
/events/EVENT_ID/meta#HOST_-TOKEN	the Settings tab of event EVENT_ID
/events/EVENT_ID/RSVPs#HOST_-TOKEN	the RSVPs tab of event EVENT_ID
/acknowledgements	honor to whom honor is due
/legalese/im	the imprint or masthead[246]

[246] https://translate.berlin/blog/impressum-uebersetzen

Path	**Page**
/legalese/pp	the privacy policy[247]
/legalese/tos	the terms of service[248]
?	not found

[247]https://privacy-icons.ch/
[248]https://tosdr.org/

A Guided Tour of Play

At this point, I assume that you have skimmed through the *Getting Started*[249] section and have gone through the tutorial[250]. If you have not done so yet, read page https://www.playframework.com/documentation/latest/ScalaHome for the sake of completeness.

[249]https://www.playframework.com/documentation/latest/Introduction

[250]https://www.playframework.com/documentation/latest/HelloWorldTutorial

Persistence

```
docker volume create mongodbdata
```

Event Sourcing

MongoDB

Ever since I evaluated NoSQL database management systems (DBMS)[251] and decided to migrate Doodle[252] from MySQL[253] to MongoDB[254], the latter has been my favorite DBMS.

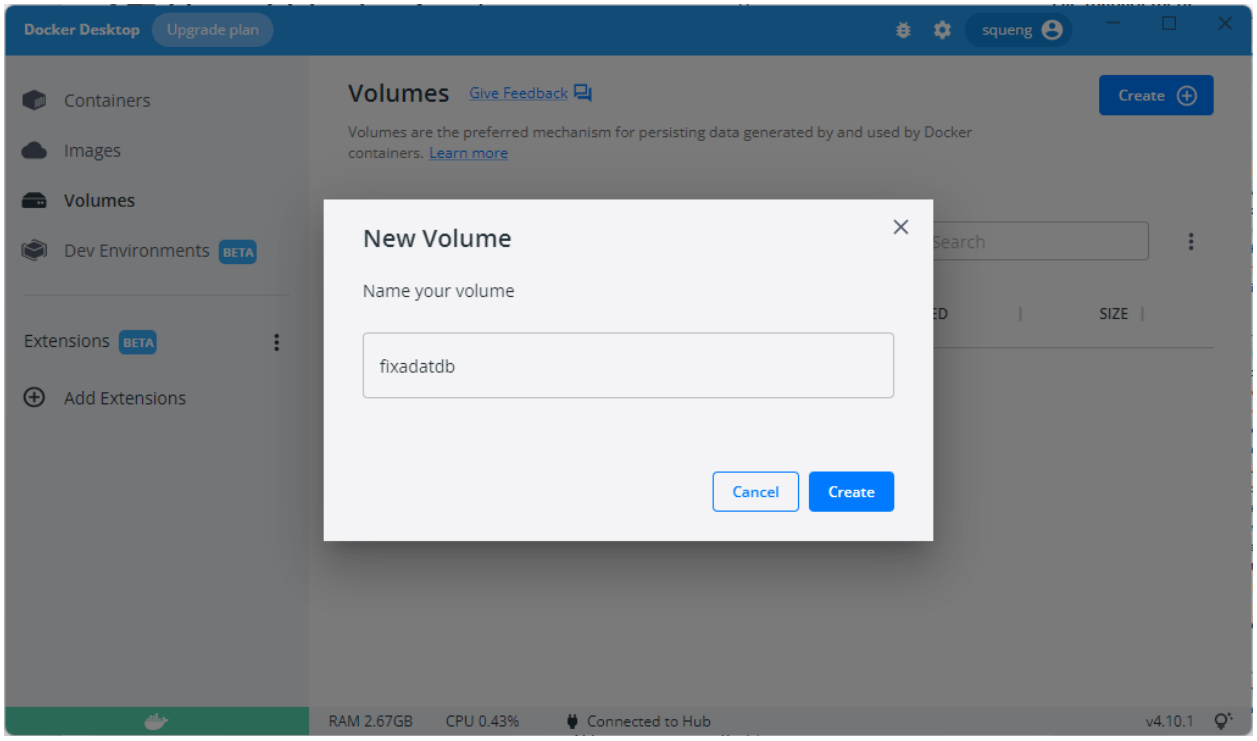
> ### Event Store
>
> At the time, MongoDB was not an obvious choice, and quite a few colleagues at least challenged my decision. And especially with respect to event sourcing, maybe it should no longer be my first choice. So if you adopt event sourcing for a production app, at least look into Event Store[a] as well as Akka Persistence[b].
>
> [a] https://www.eventstore.com/
> [b] https://doc.akka.io/docs/akka/current/typed/persistence.html

Process

Originally, I used to run MongoDB locally after downloading and installing the Community Server[255]. Alternatively, one could simply take advantage of a (free) MongoDB as a Service, from Atlas[256] for instance.

These days, I prefer to run MongoDB locally via Docker Desktop. First, I had to create a Docker volume[257] (cf. Caveats[258]). Then, I simply (re[259]-)run `docker run --name mongodbcontainer --restart unless-stopped -p 27017:27017 -v mongodbdata:/data/db -d mongo:6`.

[251] with the help of the first edition of Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken, so it must have been in late 2010 and/or early 2011
[252] https://doodle.com/
[253] https://www.mysql.com/
[254] https://www.mongodb.com/
[255] https://www.mongodb.com/try/download/community
[256] https://www.mongodb.com/atlas
[257] https://docs.docker.com/engine/reference/commandline/volume_create/
[258] https://hub.docker.com/_/mongo
[259] https://docs.docker.com/config/containers/start-containers-automatically/



Creating a volume with Docker Desktop

Driver

In order for the back-end to be able to interact with MongoDB, we need to include the Scala driver[260] in the library dependencies in `Fixadat/beapi/build.sbt` by adding the following line:

```
libraryDependencies += "org.mongodb.scala" %% "mongo-scala-driver" % "4.7.0"
```

APIs

I like to have to three browser tabs with driver API docs open, namely

- https://mongodb.github.io/mongo-java-driver/4.4/apidocs/mongodb-driver-core/index.html
- https://mongodb.github.io/mongo-java-driver/4.4/apidocs/bson/index.html
- https://mongodb.github.io/mongo-java-driver/4.4/apidocs/mongo-scala-driver/index.html

[260]https://mongodb.github.io/mongo-java-driver/4.4/driver-scala/

Configuration

Module
application.conf
insecureLocalhost.conf
delicate.conf

Code

Connection helper

note CodecRegistries.fromCodecs(new UuidCodec(UuidRepresentation.STANDARD)),

Dependency Injection

As you know,

`Fixadat/beapi/app/Module.scala`[261] instantiates the connection helper and injects MdbRepository where a Repository is required (if MongoDB is configured and not another DBMS or the DevRepository).

```scala
class Module(
    env: Environment,
    config: Configuration
) extends AbstractModule
    with Logging {
  override def configure() = {

    // https://www.playframework.com/documentation/latest/ScalaDependencyInjection#E\
ager-bindings
    bind(classOf[Mdb]).asEagerSingleton
    // we may not need it when env.mode == play.api.Mode.Dev, but it won't cause any\
 problem as long as nobody invokes Mdb.apply()

    val dbImplementationName = config.get[String]("di.db")
    val dbImplementationClass: Class[_ <: Repository] = env.classLoader
      .loadClass(dbImplementationName)
      .asSubclass(classOf[Repository])
    logger.debug(s"db implementation class is $dbImplementationClass")
    bind(classOf[Repository]).to(dbImplementationClass)
  }
}
```

[261]https://github.com/Squoss/Fixadat/blob/main/beapi/app/Module.scala

Configuration

If you have not done so yet, read the following pages:

- Configuration API[262]
- Configuration[263]
- Configuration file syntax and features[264]
- Production configuration[265]

Configuration Files

Fixadat takes advantage of both the default configuration file `application.conf`[266] and alternative configuration files[267]. If you look into folder `conf`[268], there is actually only one alternative configuration file, namely `insecureLocalhost.conf`[269]; however, `.gitignore`[270] refers to a second alternative configuration file called `delicate.conf`.

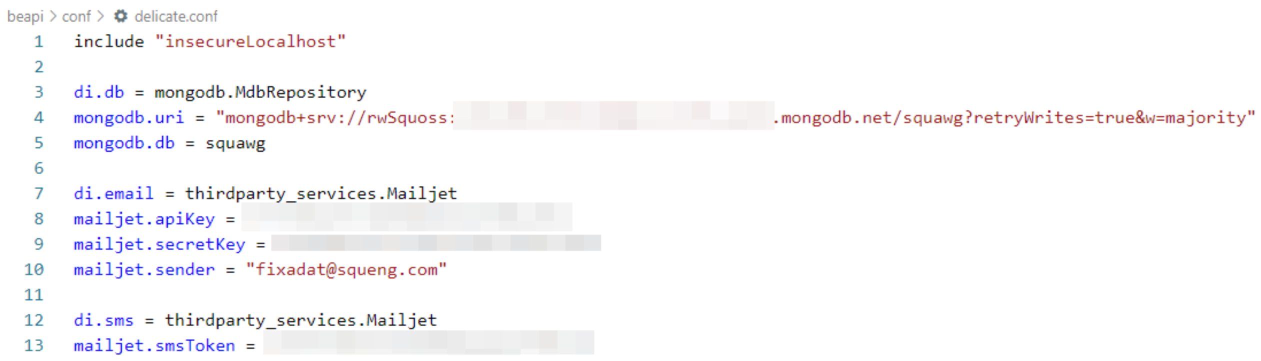
`insecureLocalhost.conf` is meant for use in run mode (i.e., development on localhost): `[Fixadat] $ run -Dconfig.file=conf/insecureLocalhost.conf`:

```
include "application"

play.filters.csrf.header.bypassHeaders.Csrf-Token = "REPLACE_CSRF_TOKEN"

di.db = dev.DevRepository
di.email = dev.DevEmail
di.sms = dev.DevSms

mongodb.uri = "mongodb://localhost:27017/fixadat"
mongodb.db = fixadat
```

[262]https://www.playframework.com/documentation/latest/ScalaConfig
[263]https://www.playframework.com/documentation/latest/Configuration
[264]https://www.playframework.com/documentation/latest/ConfigFile
[265]https://www.playframework.com/documentation/latest/ProductionConfiguration
[266]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/application.conf
[267]https://www.playframework.com/documentation/latest/ConfigFile#Specifying-an-alternative-configuration-file
[268]https://github.com/Squoss/Fixadat/main/beapi/conf
[269]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/insecureLocalhost.conf
[270]https://github.com/Squoss/Fixadat/blob/main/beapi/.gitignore

It includes the default configuration file. What makes it insecure is that it allows for a hard-coded anti-CSRF token, which is further explained in the security chapter. It configures console-based messaging and a memory-based database, which is further explained in the dependency-injection chapter; nevertheless, for the sake of convenience, is also configures a local MongoDB instance, which is further explained in the persistence chapter, such that the command for run mode does not become unwieldy: `[Fixadat] $ run -Dconfig.file=conf/insecureLocalhost.conf -Ddi.db=mongodb.MdbRepository` instead of `[Fixadat] $ run -Dconfig.file=conf/insecureLocalhost.conf -Ddi.db=mongodb.MdbRepository -Dmongodb.uri=… -Dmongodb.db=…`.

`delicate.conf` is also meant for use in run mode, but less for development on localhost and more for debugging with an actual messaging and/or database provider: `[Fixadat] $ run -Dconfig.file=conf/insecureLocalhost.conf`.



```
include "insecureLocalhost"

di.db = mongodb.MdbRepository
mongodb.uri = "mongodb+srv://rwSquoss:[redacted].mongodb.net/squawg?retryWrites=true&w=majority"
mongodb.db = squawg

di.email = thirdparty_services.Mailjet
mailjet.apiKey = [redacted]
mailjet.secretKey = [redacted]
mailjet.sender = "fixadat@squeng.com"

di.sms = thirdparty_services.Mailjet
mailjet.smsToken = [redacted]
```

Don't try this at home!

Application Secret

If you have not done so yet, read the following page:

- Configuring the application secret[271]

Session Cookie

If you have not done so yet, read the following page:

- Configuring the session cookie[272]

[271]https://www.playframework.com/documentation/latest/ApplicationSecret
[272]https://www.playframework.com/documentation/latest/SettingsSession

Logging

If you have not done so yet, read the following page:

- Configuring logging[273]

[273]https://www.playframework.com/documentation/latest/SettingsLogger

Dependency Injection

The Dependency Injection pattern[274] is a popular means of wiring objects and a natural fit for frameworks—such as Play—with their Hollywood Principle[275]. And it is particularly convenient with respect to ports & adapters.

If you have not done so yet, read the following page:

- Dependency Injection with Guice[276]

Build Scripts

First off, note how both the application build script `build.sbt`[277] (in addition to the "regular" `libraryDependencies += guice`) as well as the subproject's build script `build.sbt`[278] include the line `libraryDependencies += "com.google.inject" % "guice" % "5.1.0"`. The former does so in order to manually bump the version number for compatibility with Java 17. The latter does so in order to make the JSR 330[279] annotations (i.e., package `javax.inject`) available.

Controllers and Filters

That all four controllers as well as the one filter are defined as classes and have instances of framework types injected as necessary should come as no surprise[280].

Additionally, `ElectionsController`[281] has a domain-service instance injected while both `ReactController`[282] and `ValidationsController`[283] are defined as singletons in order for their `val`s to be cached.

[274]https://martinfowler.com/articles/injection.html
[275]https://www.informit.com/store/design-patterns-elements-of-reusable-object-oriented-9780201633610
[276]https://www.playframework.com/documentation/latest/ScalaDependencyInjection
[277]https://github.com/Squoss/Fixadat/blob/main/beapi/build.sbt
[278]https://github.com/Squoss/Fixadat/blob/main/beapi/reinraum/build.sbt
[279]https://jcp.org/en/jsr/detail?id=330
[280]https://www.playframework.com/documentation/latest/ScalaActions#Controllers-are-action-generators
[281]https://github.com/Squoss/Fixadat/blob/main/beapi/app/api/ElectionsController.scala
[282]https://github.com/Squoss/Fixadat/blob/main/beapi/app/gui/ReactController.scala
[283]https://github.com/Squoss/Fixadat/blob/main/beapi/app/api/ValidationsController.scala

A Guided Tour of React

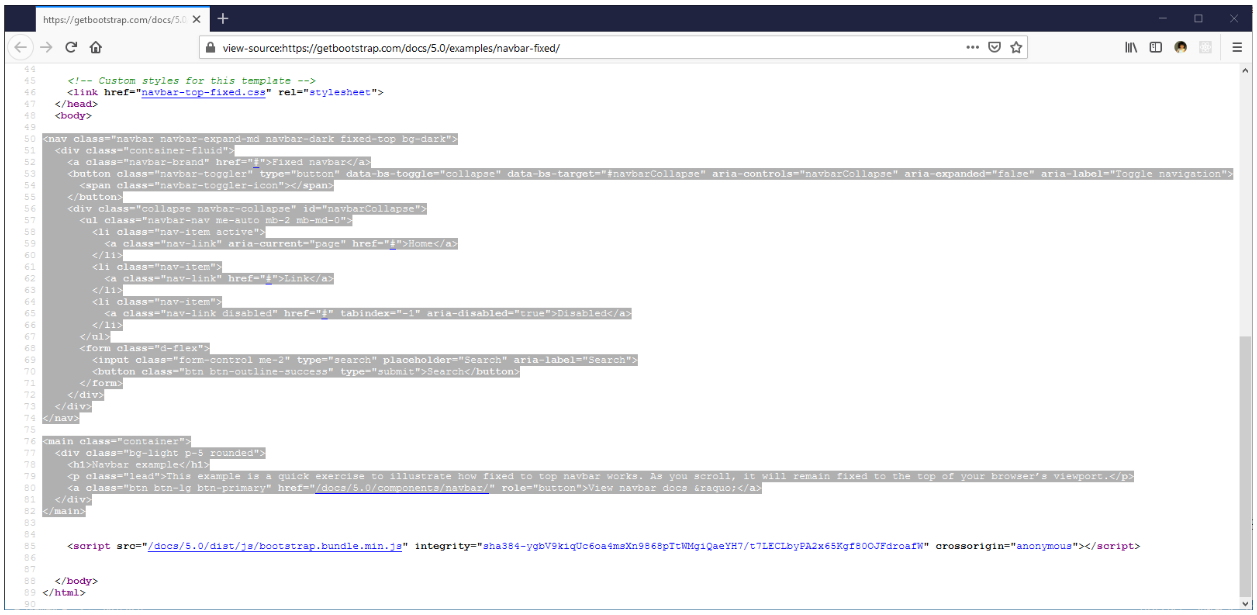
This part needs be done and fixed.

reminder to myself

- FIXME
- TODO

Template

Bootstrap features a bunch of examples[284], and our product manager and interaction designer like Navbar fixed[285] for Fixadat. To adopt it, copy the content of the body except for the script tag, wrap it in a `<React.Fragment>` tag[286], and replace the return value of `Fixadat/fegui/src/App.tsx` with it. As you know[287], you also need to replace `class=` by `className=` and `tabindex="-1"` by `tabIndex={-1}`.



https://getbootstrap.com/docs/5.0
view-source:https://getbootstrap.com/docs/5.0/examples/navbar-fixed/

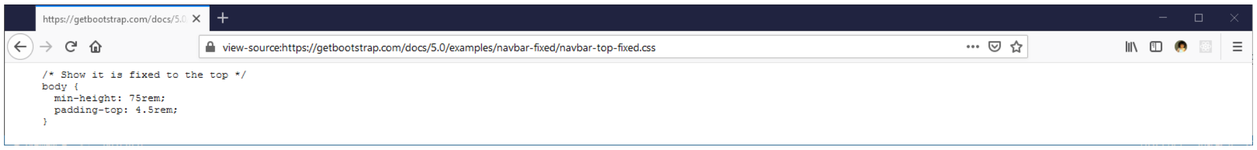
```html
    <!-- Custom styles for this template -->
    <link href="navbar-top-fixed.css" rel="stylesheet">
  </head>
  <body>

<nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Fixed navbar</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
      <ul class="navbar-nav me-auto mb-2 mb-md-0">
        <li class="nav-item active">
          <a class="nav-link" aria-current="page" href="#">Home</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#">Link</a>
        </li>
        <li class="nav-item">
          <a class="nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </li>
      </ul>
      <form class="d-flex">
        <input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">
        <button class="btn btn-outline-success" type="submit">Search</button>
      </form>
    </div>
  </div>
</nav>

<main class="container">
  <div class="bg-light p-5 rounded">
    <h1>Navbar example</h1>
    <p class="lead">This example is a quick exercise to illustrate how fixed to top navbar works. As you scroll, it will remain fixed to the top of your browser’s viewport.</p>
    <a class="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role="button">View navbar docs &raquo;</a>
  </div>
</main>


    <script src="/docs/5.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW" crossorigin="anonymous"></script>


  </body>
</html>
```

Navbar fixed HTML

Futhermore, copy its stylesheet[288] and overwrite `Fixadat/fegui/src/index.css` with it:

```css
/* source: https://getbootstrap.com/docs/5.0/examples/navbar-fixed/navbar-top-fixed.\
css */
body {
  min-height: 75rem;
  padding-top: 4.5rem;
}
```

[284] https://getbootstrap.com/docs/5.0/examples/
[285] https://getbootstrap.com/docs/5.0/examples/navbar-fixed/
[286] https://reactjs.org/docs/fragments.html
[287] from https://reactjs.org/docs/introducing-jsx.html#specifying-attributes-with-jsx
[288] https://getbootstrap.com/docs/5.0/examples/navbar-fixed/navbar-top-fixed.css


https://getbootstrap.com/docs/5.0/
view-source:https://getbootstrap.com/docs/5.0/examples/navbar-fixed/navbar-top-fixed.css

```
/* Show it is fixed to the top */
body {
  min-height: 75rem;
  padding-top: 4.5rem;
}
```

Navbar fixed CSS

Finally, delete `Fixadat/fegui/src/App.css` and `Fixadat/fegui/src/logo.svg` as well as their imports in `Fixadat/fegui/src/App.tsx`. The latter now looks as follows after having added a `<footer>` tag[289] consisting of a simple card[290] fixed to the bottom[291]:

```
import React from 'react';

function App() {
  return (
    <React.Fragment>
      <header>
        <nav className="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
          <div className="container-fluid">
            <a className="navbar-brand" href="#">Fixed navbar</a>
            <button className="navbar-toggler" type="button" data-bs-toggle="collaps\
e" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="fa\
lse" aria-label="Toggle navigation">
              <span className="navbar-toggler-icon"></span>
            </button>
            <div className="collapse navbar-collapse" id="navbarCollapse">
              <ul className="navbar-nav me-auto mb-2 mb-md-0">
                <li className="nav-item active">
                  <a className="nav-link" aria-current="page" href="#">Home</a>
                </li>
                <li className="nav-item">
                  <a className="nav-link" href="#">Link</a>
                </li>
                <li className="nav-item">
                  <a className="nav-link disabled" href="#" tabIndex={-1} aria-disab\
led="true">Disabled</a>
                </li>
              </ul>
              <form className="d-flex">
                <input className="form-control me-2" type="search" placeholder="Sear\
ch" aria-label="Search" />
                <button className="btn btn-outline-success" type="submit">Search</bu\
```

[289]https://developer.mozilla.org/en-US/docs/Web/HTML/Element/footer
[290]https://getbootstrap.com/docs/5.0/components/card/
[291]https://getbootstrap.com/docs/5.0/helpers/position/#fixed-bottom

```
tton>
              </form>
            </div>
          </div>
        </nav>
      </header>

      <main className="container">
        <div className="bg-light p-5 rounded">
          <h1>Navbar example</h1>
          <p className="lead">This example is a quick exercise to illustrate how fix\
ed to top navbar works. As you scroll, it will remain fixed to the top of your brows\
er's viewport.</p>
          <a className="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" \
role="button">View navbar docs &raquo;</a>
        </div>
      </main>

      <footer className="fixed-bottom">
        <div className="card text-end">
          <div className="card-body">
            Copyright &copy; <time dateTime="2021">2021</time> Squeng AG
          </div>
        </div>
      </footer>
    </React.Fragment>
  );
}

export default App;
```

While the navigation (at least for the most part) and the footer are meant to look the same on every path, the main section must reflect the current path.

Delete the main section's content and add the following function as `Abode.tsx` in `Fixadat/fegui/src`:

```
function Abode() {
  return (
    <div className="bg-light p-5 rounded">
      <h1>répondez s'il vous plaît</h1>
      <p className="lead">Use your favorite off-line &amp; on-line channels for invi\
tations and let invitees conveniently RSVP in one place.</p>
      <a className="btn btn-lg btn-primary" href="/docs/5.0/components/navbar/" role\
="button">Create a basic page for RSVPing  &raquo;</a>
      <p>It's free of charge and requires no signing up.</p>
    </div>
  );
}

export default Abode;
```

Add the following function as `NotFound.tsx` in `Fixadat/fegui/src`:

```
import { Link } from 'react-router-dom';

function NotFound() {
  return (
    <div className="alert alert-info" role="alert">
      <h4 className="alert-heading">Not Found</h4>
      <p>The page that you have requested has not been found.</p>
      <hr />
      <p className="mb-0"><Link to="/" className="alert-link"><i className="bi-house\
"></i></Link></p>
    </div>
  );
}

export default NotFound;
```

And because we cannot develop all React components at once, add the following function as `ToDo.tsx` in `Fixadat/fegui/src`:

```
function ToDo() {
  return (
    <mark>To Do</mark>
  );
}

export default ToDo;
```

Now we have all we need to update the template above with the correct links. To cut a long story short, `Fixadat/fegui/src/App.tsx` now looks as follows after additionally having added links to Squeng[292] and GitHub[293]:

```
import React from 'react';
import { Link, NavLink, Redirect, Route, Switch } from 'react-router-dom';

import Abode from './Abode';
import NotFound from './NotFound';
import ToDo from './ToDo';

function App() {
  return (
    <React.Fragment>
      <header>
        <nav className="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
          <div className="container-fluid">
            <Link className="navbar-brand" to="/">Fixadat</Link> <a className="navba\
r-brand" href="https://io.squeng.com/abode/" target="Squeng"><small>Squeng<sup>&reg;\
</sup>&nbsp;made</small></a>
            <button className="navbar-toggler" type="button" data-bs-toggle="collaps\
e" data-bs-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="fa\
lse" aria-label="Toggle navigation">
              <span className="navbar-toggler-icon"></span>
            </button>
            <div className="collapse navbar-collapse" id="navbarCollapse">
              <ul className="navbar-nav me-auto mb-2 mb-md-0">
                <li className="nav-item">
                  <NavLink className="nav-link" exact={true} activeClassName="active\
" aria-current="page" to="/"><i className="bi-house"></i></NavLink>
                </li>
                <li className="nav-item">
                  <NavLink className="nav-link" activeClassName="active" to="/acknow\
```

[292]https://io.squeng.com/abode/
[293]https://github.com/Squoss/Fixadat

```
ledgements">Acknowledgements</NavLink>
                </li>
                <li className="nav-item dropdown">
                  <a className="nav-link dropdown-toggle" href="#" id="legalese" rol\
e="button" data-bs-toggle="dropdown" aria-expanded="false">Legalese</a>
                  <ul className="dropdown-menu" aria-labelledby="legalese">
                    <li><NavLink className="dropdown-item" activeClassName="disabled\
" to="/legalese/im">Imprint / Masthead</NavLink></li>
                    <li><NavLink className="dropdown-item" activeClassName="disabled\
" to="/legalese/pp">Privacy Policy</NavLink></li>
                    <li><NavLink className="dropdown-item" activeClassName="disabled\
" to="/legalese/tos">Terms of Service</NavLink></li>
                  </ul>
                </li>
                <li className="nav-item">
                  <a className="nav-link disabled" href="#" tabIndex={-1} aria-disab\
led="true">Prices</a>
                </li>
              </ul>
              <a className="btn btn-secondary" href="https://github.com/Squoss/Fixad\
at" target="GitHub"><i className="bi-github"></i></a>
            </div>
          </div>
        </nav>
      </header>

      <main className="container">
        <Switch>
          <Route exact path="/" component={Abode} />
          <Route path="/events/:event" component={ToDo} />
          <Route path="/acknowledgements" component={ToDo} />
          <Route exact path="/legalese">
            <Redirect to="/legalese/im" />
          </Route>
          <Route path="/legalese/im" component={ToDo} />
          <Route path="/legalese/pp" component={ToDo} />
          <Route path="/legalese/tos" component={ToDo} />
          {/* when none of the above match, <NotFound> will be rendered */}
          <Route component={NotFound} />
        </Switch>
      </main>

      <footer className="fixed-bottom">
```

```
        <div className="card text-end">
          <div className="card-body">
            Copyright &copy; <time dateTime="2021">2021</time> Squeng AG
          </div>
        </div>
      </footer>
    </React.Fragment>
  );
}

export default App;
```

Internationalization and Localization

React in general and Create React App in particular do not support internationalization (118n) and localization (l10n) out of the box. However, there are several third-party i18n/l10n libraries that work with React (Mozilla's research project Fluent[294], for instance), some even exclusively.

As you know[295], Play supports i18n/l10n out of the box. Because it is based on Java's time-tested mechanisms and because we also need localized texts in the back-end (e.g, when sending e-mail) but want to avoid redundancy, we are taking an approach inspired by the Play JsMessages library[296].

Schnittstelle

Back-end

This section requires familiarity with Java's Locale[a], PropertyResourceBundle[b], and MessageFormat[c].

[a]https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/Locale.html
[b]https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/PropertyResourceBundle.html
[c]https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/text/MessageFormat.html

Supported Locales

Since we want English (en) to be the default/fallback locale, we use the existing `Fixadat/beapi/conf/messages`[297] for it and add `Fixadat/beapi/conf/messages.de`[298] for German (de). Then we list the supported locales by adding the key-value pair `play.i18n.langs = ["en", "de"]` in `Fixadat/beapi/conf/application.conf`[299]. Note how both messages files start with sort of a self-reference: `locale = …`; its meaning is going to become clear below.

[294]https://github.com/projectfluent/fluent.js/tree/master/fluent-react
[295]from https://www.playframework.com/documentation/latest/ScalaI18N
[296]https://github.com/julienrf/play-jsmessages
[297]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/messages
[298]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/messages.de
[299]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/application.conf

Switching the Locale

As you know[300], Play does a good job of interpreting a HTTP request's Accept-Language header. Nevertheless, we want to allow for explicitly switching the locale by the user. One approach is to have a controller react to a change-locale request, switch the locale, and respond with a redirection (e.g., back to the page the user was already on). Another approach is to have a filter intercept all requests and determine whether the locale needs switching before the request is handled any further. We take the latter approach and are going to see below what that means for the front-end.

Add the following class to `Fixadat/beapi/app/filters` and then activate it by adding the key-value pair `play.filters.enabled += filters.MessagesFilter` in `Fixadat/beapi/conf/application.conf`[301]:

```scala
package filters

import akka.stream.Materializer
import play.api.Configuration
import play.api.Logging
import play.api.i18n.I18nSupport
import play.api.i18n.Lang
import play.api.mvc.Cookie
import play.api.mvc.Filter
import play.api.mvc.RequestHeader
import play.api.mvc.Result
import play.api.routing.HandlerDef
import play.api.routing.Router

import javax.inject.Inject
import scala.concurrent.ExecutionContext
import scala.concurrent.Future

class MessagesFilter @Inject() (implicit
    val mat: Materializer,
    ec: ExecutionContext,
    config: Configuration
) extends Filter {

  def apply(
      nextFilter: RequestHeader => Future[Result]
  )(requestHeader: RequestHeader): Future[Result] = {

```

[300] from https://www.playframework.com/documentation/latest/ScalaContentNegotiation#Language
[301] https://github.com/Squoss/Fixadat/blob/main/beapi/conf/application.conf

```
    // cannot use I18nSupport above (this is a filter, not a controller) and therefo\
re not result.withLang below
    // https://www.playframework.com/documentation/latest/ScalaI18N#Language-Cookie-\
Support
    // for future reference: https://www.playframework.com/documentation/latest/api/\
scala/play/api/i18n/MessagesApi.html#setLang(result:play.api.mvc.Result,lang:play.ap\
i.i18n.Lang):play.api.mvc.Result
    val cookieName = config
      .getOptional[String]("play.i18n.langCookieName")
      .getOrElse("PLAY_LANG")
    val query = requestHeader.queryString
    val newLocale =
      query.get("locale").flatMap(_.headOption.flatMap(Lang.get(_)))
    if (newLocale.isEmpty) {
      nextFilter(requestHeader).map { result => result }
    } else {
      nextFilter(requestHeader.withTransientLang(newLocale.get)).map { result =>
        result.withCookies(Cookie(cookieName, newLocale.get.code))
      }
    }
  }
}
```

Serving the Localizations

routes

```
  def jsonMessages = Action { implicit request =>
    val lang = request.lang
    val default = messagesApi.messages.get("default").getOrElse(Map())
    val language =
      messagesApi.messages.get(Lang(lang.language).code).getOrElse(Map())
    val country = messagesApi.messages
      .get(Lang(lang.language, lang.country).code)
      .getOrElse(Map())
    val script = messagesApi.messages
      .get(Lang(lang.language, lang.country, lang.script).code)
      .getOrElse(Map())
    val variant = messagesApi.messages
      .get(Lang(lang.language, lang.country, lang.script, lang.variant).code)
      .getOrElse(Map())
    Ok(Json.toJson(default ++ language ++ country ++ script ++ variant))
  }
```

Front-end

In addition to what is listed in the Guided Tours chapter, this section requires familiarity with using contexts[a] and the useContext[b] hook.

[a] https://reactjs.org/docs/context.html
[b] https://reactjs.org/docs/hooks-reference.html#usecontext

Using a Localization Context

```
import React from "react";

export interface Localizations {
  [key: string]: string;
}

export const l10nContext = React.createContext<Localizations>({});
```

Fetching and Providing the Localizations

```
import React, { useEffect, useState } from "react";
import App from "./App";
import { get } from "./fetchJson";
import { l10nContext, Localizations } from "./l10nContext";


function I18nApp(props: {}) {
  console.log("I18nApp props: " + JSON.stringify(props));

  const [localizations, setLocalizations] = useState<Localizations>({});

  useEffect(() => {
    const fetchLocalizations = () => get<Localizations>("/jsonMessages", "").then(re\
sponseJson => {
      console.debug(responseJson.status);
      console.debug(responseJson.parsedBody);
      setLocalizations(responseJson.parsedBody!);
    }).catch(error => console.error(`failed to get time zones: ${error}`));

    fetchLocalizations();
  }, []);

  return (
    <React.Fragment>
      {localizations === {} ? (
        <div className="spinner-border" role="status">
          <span className="visually-hidden">Loading localizations …</span>
        </div>
      ) : (
        <l10nContext.Provider value={localizations}>
          <App />
        </l10nContext.Provider>
      )}
    </React.Fragment>
  );
}

export default I18nApp;
```

Using the Localizations

```
import { useContext } from 'react';
import { Link } from 'react-router-dom';
import { l10nContext } from "./l10nContext";

function NotFound(props: {}) {
  console.log("NotFound props: " + JSON.stringify(props));

  const localizations = useContext(l10nContext);

  return (
    <div className="alert alert-info" role="alert">
      <h4 className="alert-heading">{localizations['notFound.title']}</h4>
      <p>{localizations['notFound.page']}</p>
      <hr />
      <p className="mb-0"><Link to="/" className="alert-link"><i className="bi bi-ho\
use"></i></Link></p>
    </div>
  );
}

export default NotFound;
```

Switching the Locale

```
import { Modal } from "bootstrap";
import React, { useContext, useEffect } from 'react';
import { Link, NavLink, Redirect, Route, Switch, useLocation } from 'react-router-do\
m';
import Abode from './Abode';
import EventComponent from './EventComponent';
import { l10nContext } from './l10nContext';
import NotFound from './NotFound';
import ToDo from './ToDo';


function App(props: {}) {
  console.log("App props: " + JSON.stringify(props));

  const localizations = useContext(l10nContext);

  const location = useLocation();
  let locationString = location.pathname;
  locationString += "?";
  new URLSearchParams(location.search).forEach((v, k) => locationString += k !== "lo\
cale" ? `${k}=${v}&` : "");
  locationString += "locale=NEWLOCALE";
  locationString += location.hash;
```

```
            <div className="dropdown">
              <button className="btn btn-outline-primary dropdown-toggle" type="butt\
on" id="language" data-bs-toggle="dropdown" aria-expanded="false"><i className="bi b\
i-globe"></i></button>
              <ul className="dropdown-menu" aria-labelledby="language">
                <li><a className={localizations['locale'] === "de" ? "dropdown-item \
disabled" : "dropdown-item"} href={locationString.replace("NEWLOCALE", "de")}>Deutsc\
h</a></li>
                <li><a className={localizations['locale'] === "en" ? "dropdown-item \
disabled" : "dropdown-item"} href={locationString.replace("NEWLOCALE", "en")}>Englis\
h</a></li>
              </ul>
            </div>
```

`http://localhost:9000/?locale=en` or `http://localhost:9000/?locale=de`
`http://localhost:9000/?locale=de-CH` defaults to German and `http://localhost:9000/?locale=fr-CA` defaults to English

Privacy

https://www.heise.de/hintergrund/Softwareentwicklung-und-Datenschutz-wie-passt-das-zusammen-6155870.html?seite=all

https://www.springer.com/de/book/9783662630860

https://dpunkt.de/produkt/cloud-computing-nach-der-datenschutz-grundverordnung/

Legalese

Consent Management: https://usercentrics.com/

Hello, world!

This part needs be done and fixed.

Like the first part and unlike the previous part, this entire part is pretty much independent of the domain.

README

- Configuration
  - Configuring the application secret[302]
- Deploying your application
  - Using Play in production[303]
  - Deploying your application[304]
  - Production configuration[305]

[302]https://www.playframework.com/documentation/latest/ApplicationSecret
[303]https://www.playframework.com/documentation/latest/Production
[304]https://www.playframework.com/documentation/latest/Deploying
[305]https://www.playframework.com/documentation/latest/ProductionConfiguration

Going Public

index.html

Before we publish the Web app, we should change the title and the description in `Fixadat/fegui/public/index.html`. While we are at it, we could also change the favicon and update `Fixadat/fegui/public/manifest.json`; services such as Favicon Generator[306] can help with both. `Fixadat/fegui/public/index.html` now looks as follows:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="description" content="Squeng's RSVP app" />
    <!-- generated with https://www.ionos.com/tools/favicon-generator -->
    <link rel="apple-touch-icon" sizes="57x57" href="%PUBLIC_URL%/apple-icon-57x57.p\
ng" />
    <link rel="apple-touch-icon" sizes="60x60" href="%PUBLIC_URL%/apple-icon-60x60.p\
ng" />
    <link rel="apple-touch-icon" sizes="72x72" href="%PUBLIC_URL%/apple-icon-72x72.p\
ng" />
    <link rel="apple-touch-icon" sizes="76x76" href="%PUBLIC_URL%/apple-icon-76x76.p\
ng" />
    <link rel="apple-touch-icon" sizes="114x114" href="%PUBLIC_URL%/apple-icon-114x1\
14.png" />
    <link rel="apple-touch-icon" sizes="120x120" href="%PUBLIC_URL%/apple-icon-120x1\
20.png" />
    <link rel="apple-touch-icon" sizes="144x144" href="%PUBLIC_URL%/apple-icon-144x1\
44.png" />
    <link rel="apple-touch-icon" sizes="152x152" href="%PUBLIC_URL%/apple-icon-152x1\
52.png" />
    <link rel="apple-touch-icon" sizes="180x180" href="%PUBLIC_URL%/apple-icon-180x1\
80.png" />
    <link rel="icon" type="image/png" sizes="192x192"  href="%PUBLIC_URL%/android-ic\
on-192x192.png" />
```

[306] https://www.ionos.com/tools/favicon-generator

```
    <link rel="icon" type="image/png" sizes="32x32" href="%PUBLIC_URL%/favicon-32x32\
.png" />
    <link rel="icon" type="image/png" sizes="96x96" href="%PUBLIC_URL%/favicon-96x96\
.png" />
    <link rel="icon" type="image/png" sizes="16x16" href="%PUBLIC_URL%/favicon-16x16\
.png" />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <meta name="msapplication-TileColor" content="#ffffff" />
    <meta name="msapplication-TileImage" content="%PUBLIC_URL%/ms-icon-144x144.png" \
/>
    <!-- https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta/name/theme-c\
olor -->
    <meta name="theme-color" media="(prefers-color-scheme: light)" content="white">
    <meta name="theme-color" media="(prefers-color-scheme: dark)" content="black">
    <title>Fixadat</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Note that we replaced both Create React App's and Favicon Generator's `theme-color` by a media-dependent one.

build.sbt

Two Become One 2

You do not need to be familiar with Docker[307], let alone understand the Docker instructions below in detail. This section is merely meant to shed light on how two independent (sub-)projects during development can result in one self-contained system[308] during production.

Dockerfiles contain the instructions for building a Docker image, which is instantiated as a Docker container at runtime. Our Dockerfile takes advantage of Docker's support for multi-stage builds[309].

The first stage builds the front-end:

```
FROM node:16 as react

WORKDIR /squeng/fixadat

COPY fegui/.env ./
# COPY fegui/.npmrc ./
COPY fegui/package*.json ./
COPY fegui/tsconfig.json ./
# https://docs.npmjs.com/cli/v7/commands/npm-ci
RUN npm ci

COPY fegui/public ./public
COPY fegui/src ./src
# https://create-react-app.dev/docs/adding-custom-environment-variables#linux-macos-\
bash
RUN INLINE_RUNTIME_CHUNK=false npm run build
```

As you know[310], Play can serve static, public assets. The second stage copies the front-end artifacts built during the first stage into the back-end as such assets and builds it:

[307]https://www.docker.com/
[308]https://scs-architecture.org/
[309]https://docs.docker.com/develop/develop-images/multistage-build/
[310]from https://www.playframework.com/documentation/latest/Assets and https://www.playframework.com/documentation/latest/AssetsOverview

```
FROM hseeberger/scala-sbt:17.0.1_1.6.1_2.13.8 as play

WORKDIR /squeng/fixadat

COPY beapi/app ./app
COPY beapi/conf ./conf
COPY beapi/project ./project
COPY beapi/public ./public
COPY beapi/reinraum ./reinraum
COPY beapi/build.sbt ./
COPY --from=react /squeng/fixadat/build ./public/build
RUN sbt stage
```

The thrid stage copies the artifacts integrated and built during the second stage and builds the actual Docker image:

```
FROM openjdk:17-slim

WORKDIR /squeng/fixadat

COPY --from=play /squeng/fixadat/target/universal/stage ./target/universal/stage

RUN groupadd -r gruppe && useradd --no-log-init -r -g gruppe benutzer
RUN chown -R benutzer:gruppe /squeng
USER benutzer

EXPOSE 8080
CMD ["target/universal/stage/bin/fixadat", "-Dpidfile.path=play.pid", "-Dhttp.port=8\
080"]
```

That is all there is to it.

Alternatives

There are both alternatives and variants to this set-up. One alternative is to integrate the front-end and the back-end from the very beginning (like ASP.NET Core's React project template[311], for instance). Another one is to keep the front-end and the back-end separate by deploying them independently (and worrying more about CORS[312] than CSRF). A variant is not to use Docker's multi-stage build but to build & combine the front-end and the back-end "directly" in your project's pipeline[313] first and then package it in a Docker image (if you are using Docker at all). And so on and so forth – one size does not fit all.

[311]https://docs.microsoft.com/en-us/aspnet/core/client-side/spa/react
[312]https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
[313]https://leanpub.com/cd-pipelines

Building Images Locally

In order to test building the Docker image, you can run `docker build -t="squeng/fixadat" .` in `Fixadat`. Before doing so, you may want to optimize the local build time by adding the following as `Fixadat/.dockerignore`:

```
# https://docs.docker.com/engine/reference/builder/#dockerignore-file

# the following lists need not be exhaustive but should at least include the worst o\
ffenders (i.e., largest directories)

feuwag/build
feuwag/node_modules

**/.bloop
**/.metals
**/target
```

Running Containers Locally

In order to test the Docker container (wether the image was built locally or pulled from Docker Hub), you can run

```
docker run -p 9090:8080 -p 9494:8484 -e "APPLICATION_SECRET=SUPERSECRETAPPSECRET" -e\
 "HERE_API_KEY=SUPERSECRETAPIKEY" -e "MAILJET_API_KEY=NOTSOSECRETAPIKEY" -e "MAILJET\
_SECRET_KEY=SUPERSECRETSECRETKEY" -e "MAILJET_SMS_TOKEN=SUPERSECRETSMSTOKEN" -e "MON\
GODB_DB=fixadat" -e "MONGODB_URI=mongodb+srv://rwSquoss:SUPERSECRETPASSWORD@cluster0\
.jam64.mongodb.net/fixadat?retryWrites=true&w=majority" squeng/fixadat
```

In the example above, ports 8080 and 8484 within the container are mapped to local ports 9090 and 9494, respectively. Therefore, when visiting http://localhost:9090/ one would be redirected to https://localhost:8484/ instead of https://localhost:9494/.

Continuous Integration

In this chapter and the next two, we set our pipeline up. Since we decided to use GitHub when we set our project up, we are going to use GitHub Actions[314] for our pipeline. Note that all the version-control services mentioned at the beginning of this book provide workflow automation and/or integrate with third-party providers.

If the concept of pipelines in general or GitHub Actions in particular are new to you, you are kindly referred to Continuous Delivery Pipelines[315] and Automating Workflows with GitHub Actions[316], respectively.

When working in a team, I like to adopt trunk-based development with short-lived feature branches[317]. Thus, we test (for defects) whenever a Git commit is pushed to its feature branch and when a pull request is merged into the trunk and we scan (for security) when all the tests pass (to scan the latest codebase) and once a day (to re-scan the dependencies).

Test

Testing simply means running the tests that we could (and should) run locally as well. In `Fixadat/.github/workflows`[318], add the following as `test.yml`[319]:

```yaml
name: Test

on:
  push:
    branches: [ '**' ]
  pull_request:
    branches: [ main ]

jobs:
  npm:
    runs-on: ubuntu-latest
    steps:
```

[314] https://github.com/features/actions
[315] https://leanpub.com/cd-pipelines
[316] https://www.packtpub.com/product/automating-workflows-with-github-actions/9781800560406
[317] https://trunkbaseddevelopment.com/short-lived-feature-branches/
[318] https://github.com/Squoss/Fixadat/tree/main/.github/workflows
[319] https://github.com/Squoss/Fixadat/blob/main/.github/workflows/test.yml

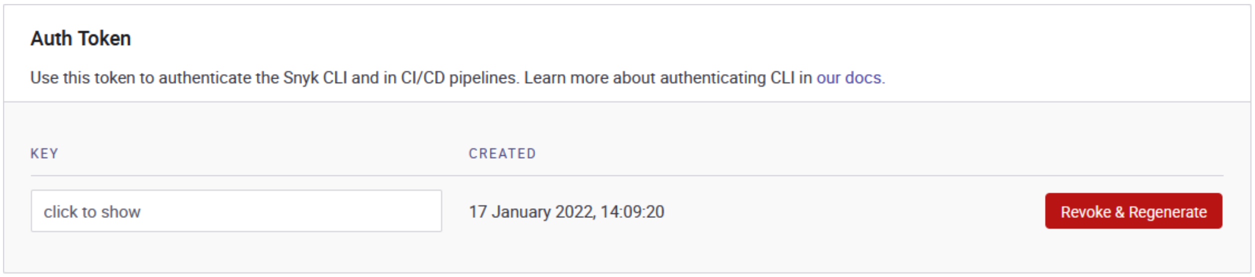
```
    - uses: actions/checkout@v2
    - name: Set up Node.js 16
      uses: actions/setup-node@v2
      with:
        node-version: '16'
        cache: 'npm'
        cache-dependency-path: fegui/package-lock.json
    - name: Clean install with npm
      run: npm ci
      working-directory: fegui
    - name: Test with npm
      run: npm test
      working-directory: fegui
  sbt:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 17
      uses: actions/setup-java@v2
      with:
        distribution: 'temurin'
        java-version: '17'
    - name: Test with sbt
      run: sbt test
      working-directory: beapi
```

Scan

As can be seen at https://github.com/Squoss/Fixadat/security/code-scanning/setup, there are quite a few code-scanning workflows to choose from. We are going to scan with Snyk[320] and SonarCloud[321].

And because I am using Visual Studio Code, I am also using the Snyk extension[322] and the SonarLint extension[323].

[320]https://github.com/snyk/actions
[321]https://github.com/SonarSource/sonarcloud-github-action
[322]https://docs.snyk.io/products/snyk-code/using-snyk-code-via-ide#vs-code-ide-plugin
[323]https://www.sonarlint.org/vscode/



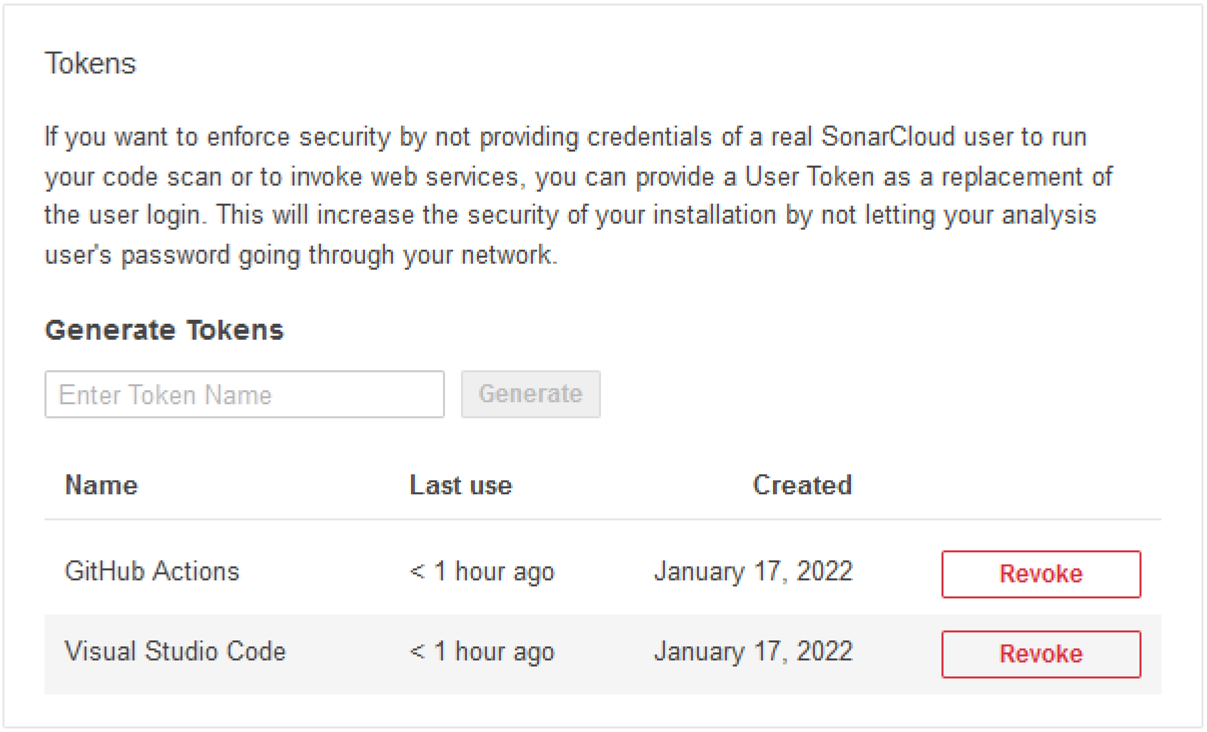
Auth Token

Use this token to authenticate the Snyk CLI and in CI/CD pipelines. Learn more about authenticating CLI in our docs.

KEY	CREATED	
click to show	17 January 2022, 14:09:20	Revoke & Regenerate

Snyk auth token

To scan with Snyk, we need to generate a token at https://app.snyk.io/account and add it as a repository secret by the name `SNYK_AUTH_TOKEN` at https://github.com/Squoss/Fixadat/settings/secrets/actions.



Tokens

If you want to enforce security by not providing credentials of a real SonarCloud user to run your code scan or to invoke web services, you can provide a User Token as a replacement of the user login. This will increase the security of your installation by not letting your analysis user's password going through your network.

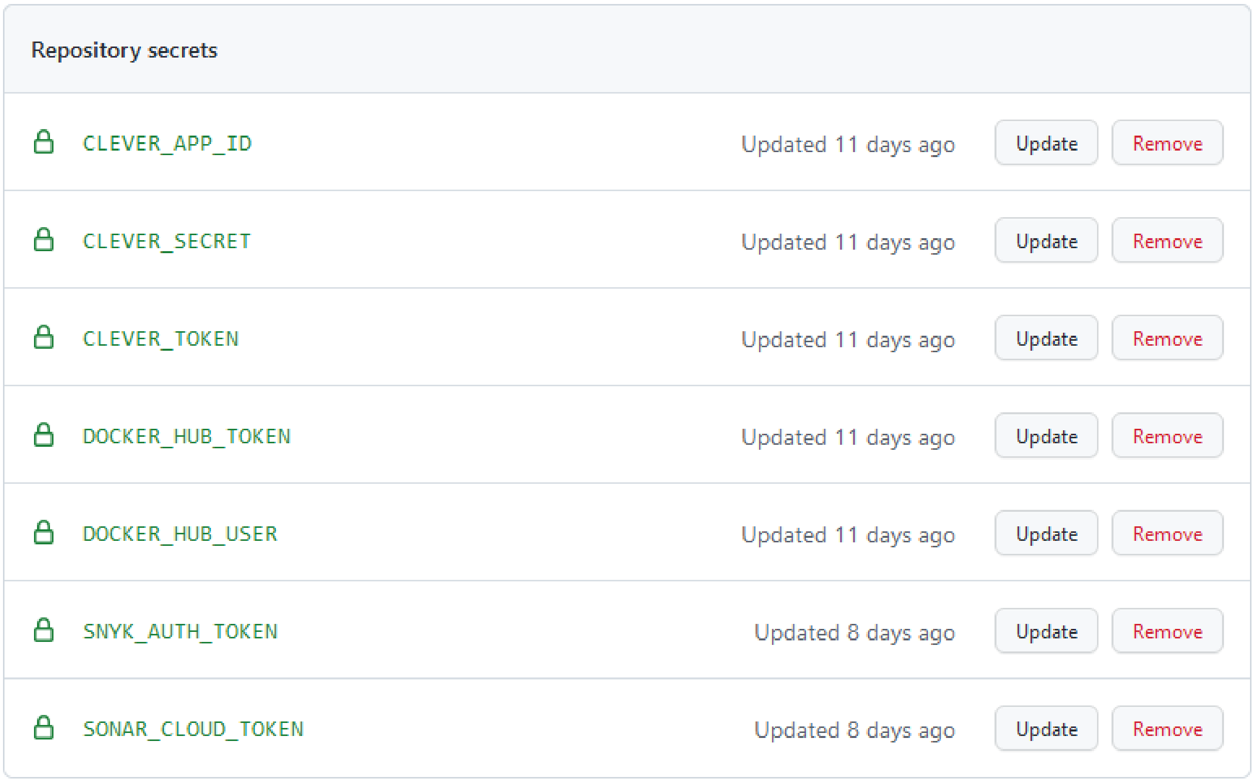
Generate Tokens

Enter Token Name | Generate

Name	Last use	Created	
GitHub Actions	< 1 hour ago	January 17, 2022	Revoke
Visual Studio Code	< 1 hour ago	January 17, 2022	Revoke

SonarCloud tokens

To scan with SonarCloud, we need to generate a token at https://sonarcloud.io/account/security/ and add it as a repository secret by the name `SONAR_CLOUD_TOKEN` at https://github.com/Squoss/Fixadat/settings/secrets/actions.



Repository secrets

CLEVER_APP_ID	Updated 11 days ago	Update	Remove
CLEVER_SECRET	Updated 11 days ago	Update	Remove
CLEVER_TOKEN	Updated 11 days ago	Update	Remove
DOCKER_HUB_TOKEN	Updated 11 days ago	Update	Remove
DOCKER_HUB_USER	Updated 11 days ago	Update	Remove
SNYK_AUTH_TOKEN	Updated 8 days ago	Update	Remove
SONAR_CLOUD_TOKEN	Updated 8 days ago	Update	Remove

GitHub repository secrets

Now we can add the following as `scan.yml`[324] in `Fixadat/.github/workflows`[325]:

```
name: Scan

on:
  workflow_run:
    workflows: ["Test"]
    types: [completed]
  schedule:
    - cron: '0 12 * * *'

jobs:
  Sonar:
    runs-on: ubuntu-latest
    if: ${{ github.event_name == 'schedule' || github.event.workflow_run.conclusion \
== 'success' }}
    steps:
      - uses: actions/checkout@v2
        with:
```

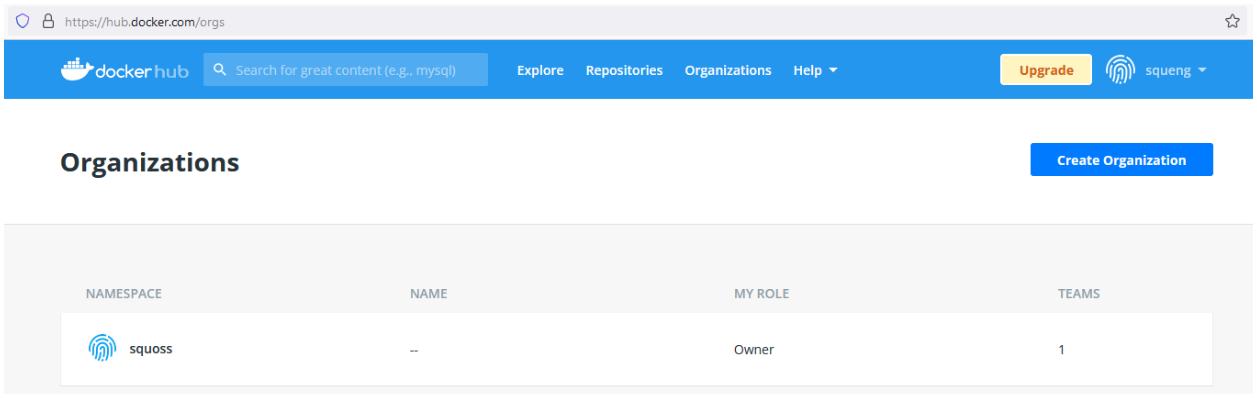
[324] https://github.com/Squoss/Fixadat/blob/main/.github/workflows/scan.yml
[325] https://github.com/Squoss/Fixadat/tree/main/.github/workflows

```
        # Disabling shallow clone is recommended for improving relevancy of reporting
          fetch-depth: 0
      - name: back-end and front-end
        uses: sonarsource/sonarcloud-github-action@master
        with:
          # https://docs.sonarcloud.io/advanced-setup/analysis-scope/#restrict-the-s\
cope-of-analysis-in-general
          args: >
            -Dsonar.organization=squoss
            -Dsonar.projectKey=Squoss_Fixadat
            -Dsonar.sources=beapi/app/,beapi/reinraum/src/main/scala/,fegui/src/
            -Dsonar.tests=beapi/test/,beapi/reinraum/src/test/scala/,fegui/src/__tes\
ts__/
            -Dsonar.exclusions=fegui/src/__tests__/**/*
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          SONAR_TOKEN: ${{ secrets.SONAR_CLOUD_TOKEN }}
  Snyk:
    runs-on: ubuntu-latest
    if: ${{ github.event_name == 'schedule' || github.event.workflow_run.conclusion \
== 'success' }}
    steps:
      - uses: actions/checkout@v2
      - name: Run Snyk to check for vulnerabilities
        # Snyk can be used to break the build when it detects security issues
        # in this case we want to upload the issues to GitHub Code Scanning
        continue-on-error: true
        id: coe
        uses: snyk/actions/node@master
        env:
          SNYK_TOKEN: ${{ secrets.SNYK_AUTH_TOKEN }}
        with:
          args: |
            --sarif-file-output=snyk.sarif
            --all-projects
      - name: Upload result to GitHub Code Scanning
        uses: github/codeql-action/upload-sarif@v1
        with:
          sarif_file: snyk.sarif
      - name: Check for failures # https://docs.github.com/en/actions/learn-github-a\
ctions/contexts#steps-context
        if: ${{ steps.coe.outcome != 'success' }}
        run: exit 1
```

Continuous Delivery

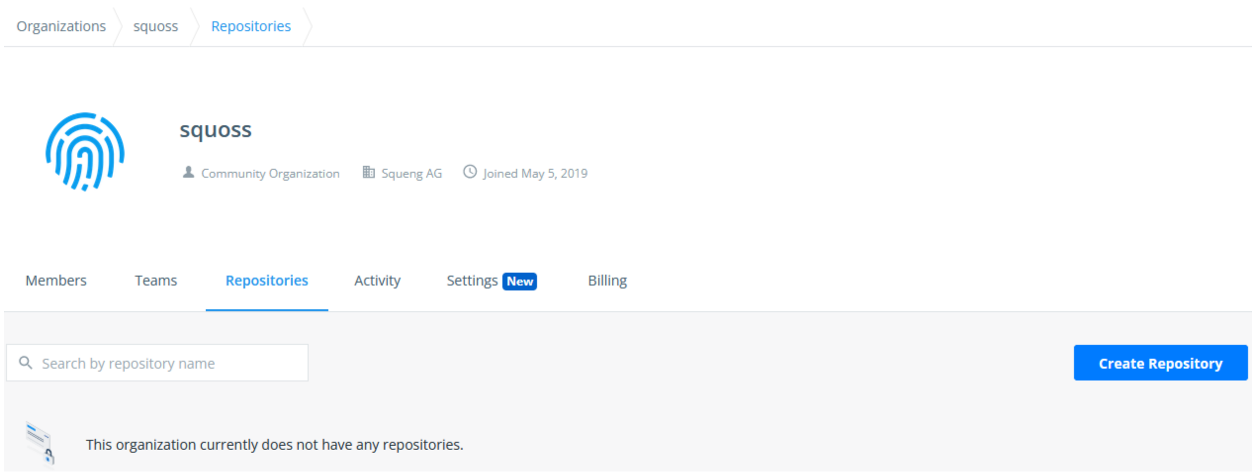
If you do not (only) provide your Web app as a service and do not want to make third parties build it themselves, you could make its Docker image available to them. A popular way to deliver Docker images is Docker Hub[326]; after all, we also use it to pull our base images and the MongoDB image. Note that there are quite a few alternatives (you should look for “container repository”, however, and not “image repository”).

The first step is to create an organization for (some of) your project(s) if you have not done so yet.



NAMESPACE	NAME	MY ROLE	TEAMS
squoss	--	Owner	1

Create Organization

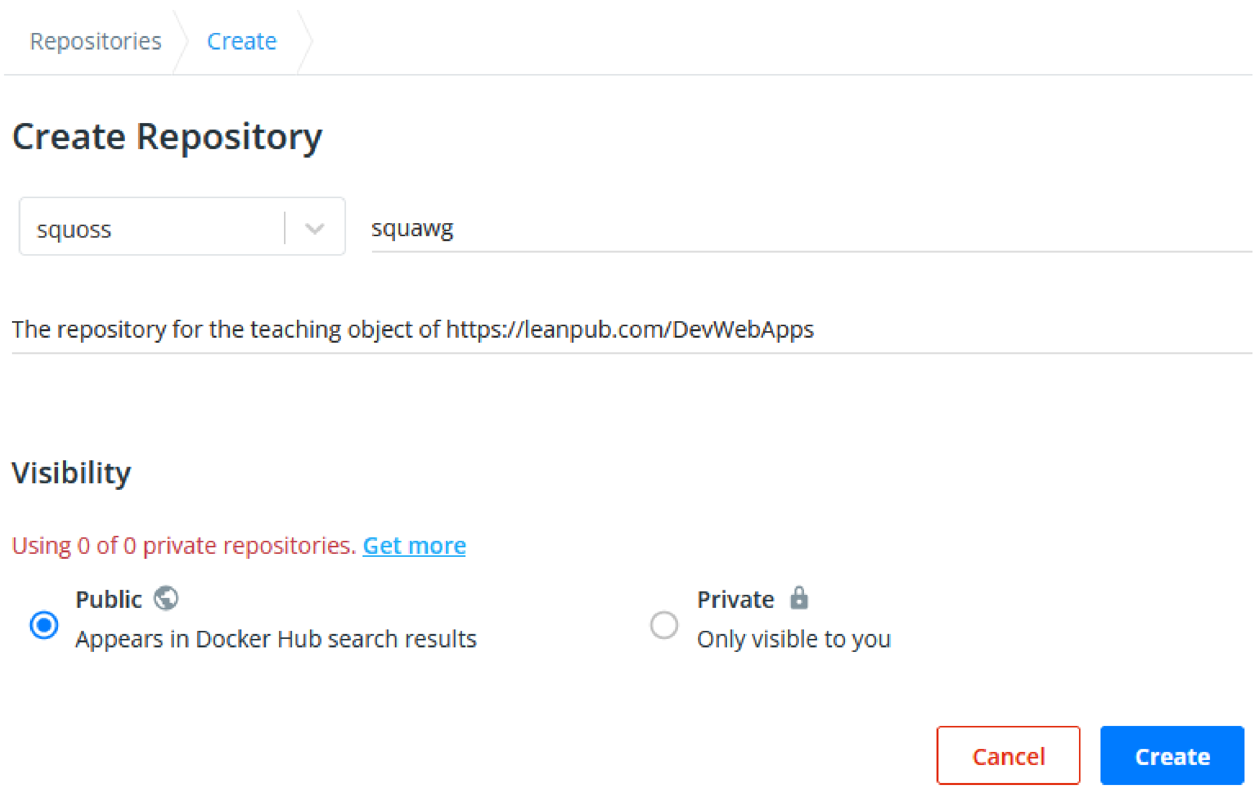


This organization currently does not have any repositories.

Created Organization

The second step is to create a repository within the organization.

[326] https://hub.docker.com/


Repositories > Create

Create Repository

squoss / squawg

The repository for the teaching object of https://leanpub.com/DevWebApps

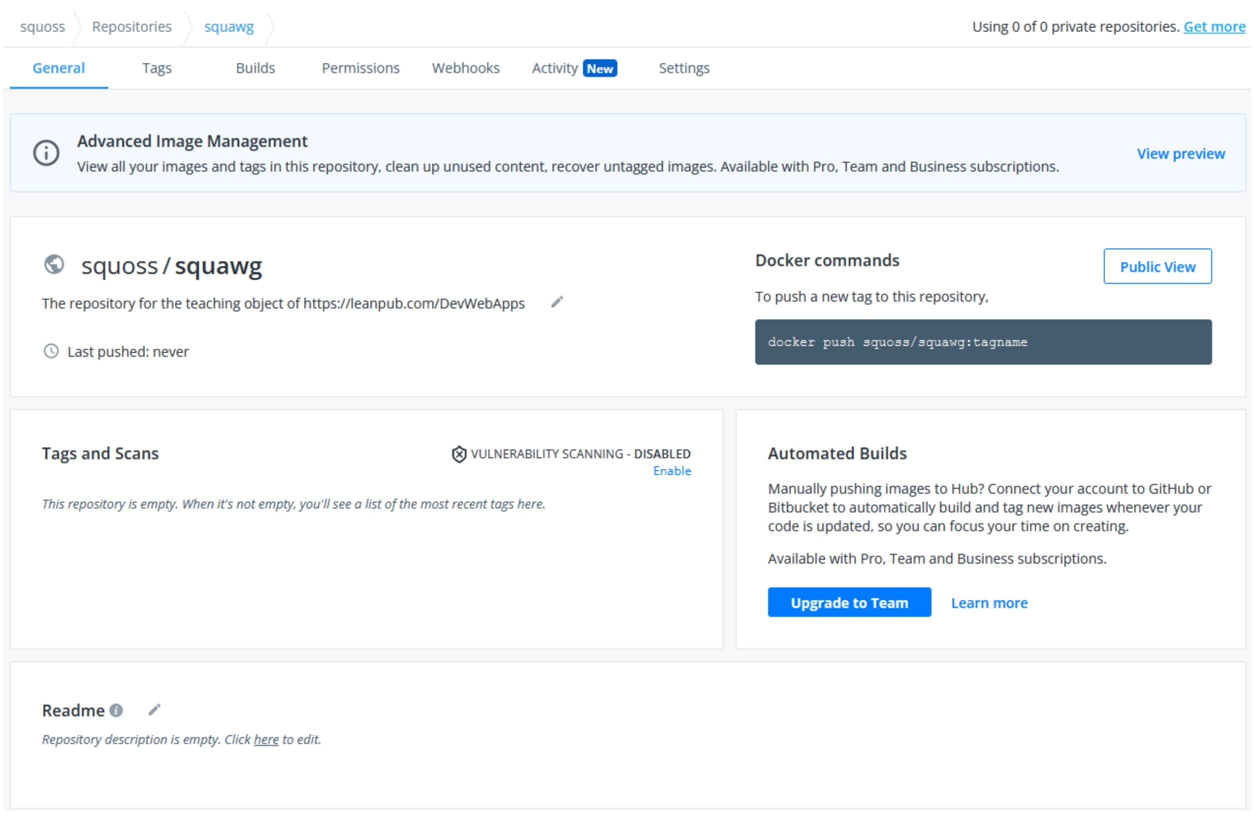
Visibility

Using 0 of 0 private repositories. Get more

[✓ Public] Appears in Docker Hub search results

Private Only visible to you

Create Repository



Advanced Image Management

View all your images and tags in this repository, clean up unused content, recover untagged images. Available with Pro, Team and Business subscriptions.

squoss / squawg

The repository for the teaching object of https://leanpub.com/DevWebApps

Last pushed: never

Docker commands

To push a new tag to this repository,

```
docker push squoss/squawg:tagname
```

Tags and Scans

This repository is empty. When it's not empty, you'll see a list of the most recent tags here.

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

Readme

Repository description is empty. Click here to edit.

Created Repository

Pull

If your project uses GitHub or Bitbucket, and depending on your Docker Hub subscription, one approach for making the delivery continuous would be to link the Docker repository to your source-code repository[327] in order for Docker Hub to pull the sources and build the image automatically[328] whenever they change. Such an approach would not necessarily be quick & dirty since it would allow for some quality assurance by automated testing[329].

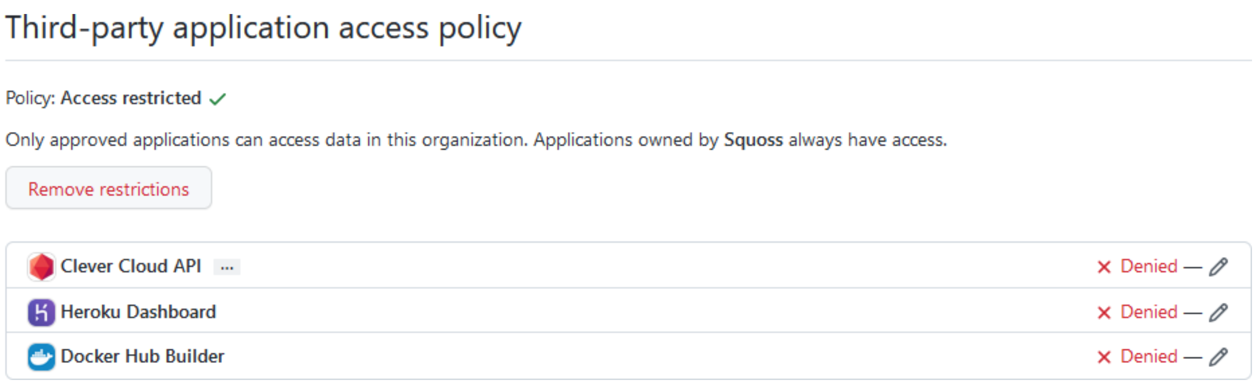
Push

The approach we are pursuing for making the delivery continuous is to extend the pipeline that we started setting up in the last chapter. If you have given the push approach above a try, you may want to clean it up & out first.

[327]https://docs.docker.com/docker-hub/builds/link-source/
[328]https://docs.docker.com/docker-hub/builds/
[329]https://docs.docker.com/docker-hub/builds/automated-testing/

Cleaning up & out

If you have not already done so,


Third-party application access policy

Policy: Access restricted ✓

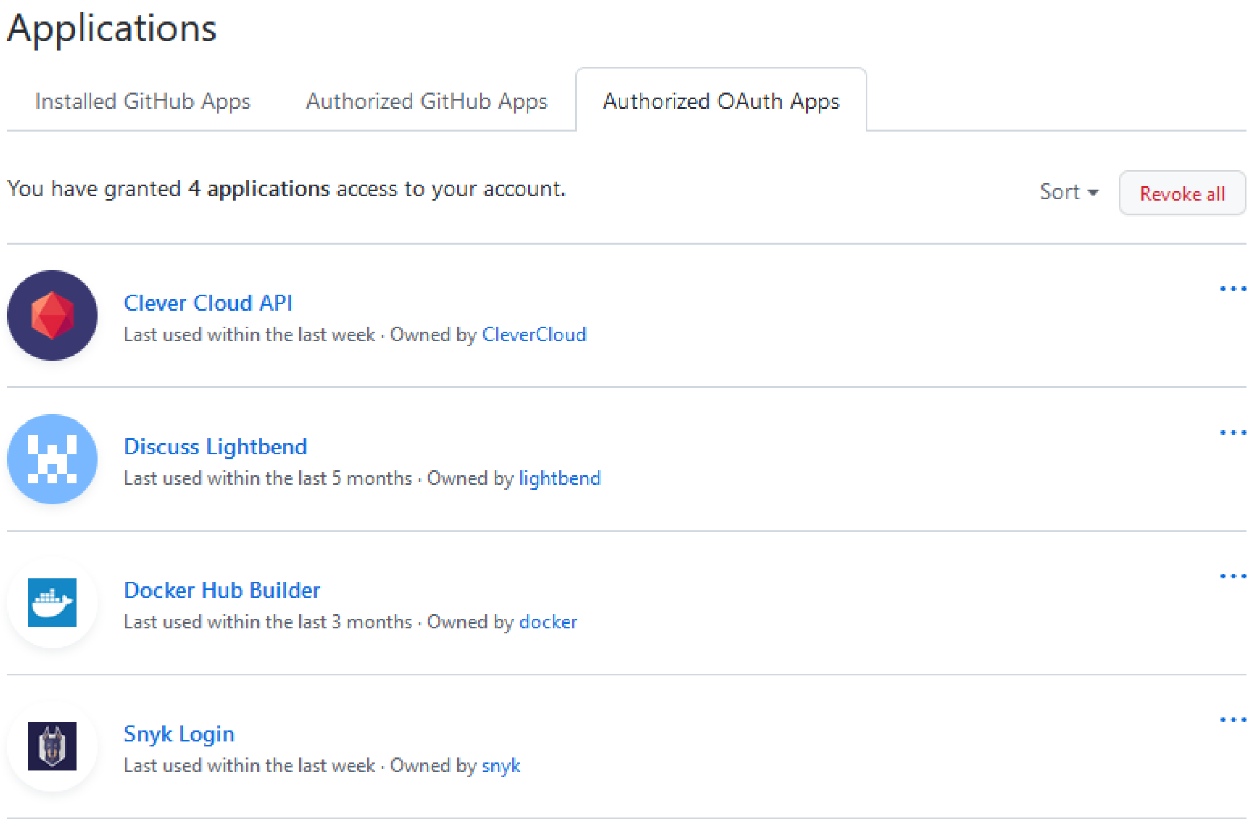
Only approved applications can access data in this organization. Applications owned by **Squoss** always have access.

Clever Cloud API	✕ Denied —
Heroku Dashboard	✕ Denied —
Docker Hub Builder	✕ Denied —

Denying access

- unlink your GitHub account from your Docker Hub account[330],
- deny Docker Hub access to your GitHub organization (at https://github.com/organizations/Squoss/settings/oauth_application_policy in Fixadat's case), and
- revoke Docker Hub's access to your personal account at https://github.com/settings/applications.

[330]https://docs.docker.com/docker-hub/builds/link-source/#unlink-a-github-user-account


Applications

Installed GitHub Apps | Authorized GitHub Apps | Authorized OAuth Apps

You have granted **4 applications** access to your account.

Clever Cloud API
Last used within the last week · Owned by CleverCloud

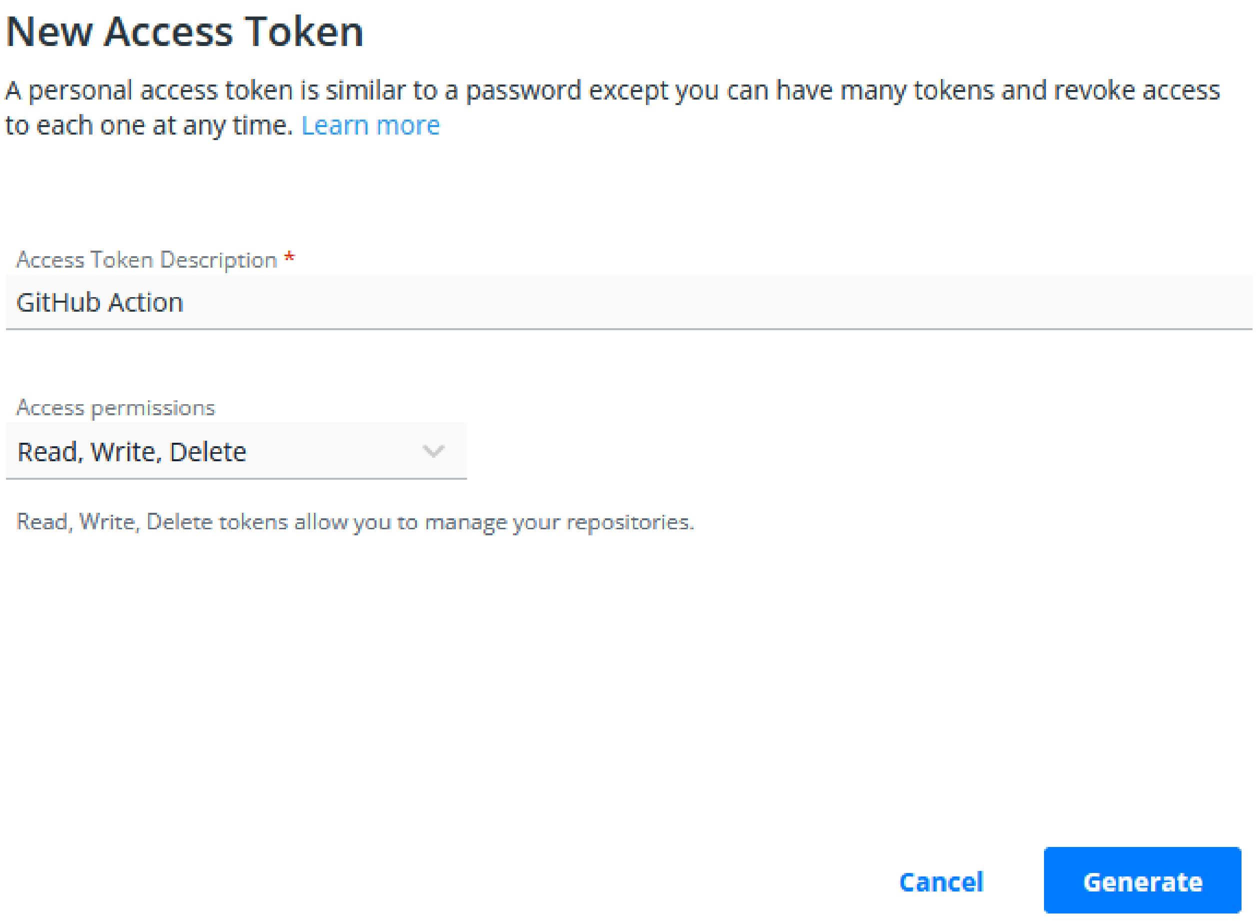
Discuss Lightbend
Last used within the last 5 months · Owned by lightbend

Docker Hub Builder
Last used within the last 3 months · Owned by docker

Snyk Login
Last used within the last week · Owned by snyk

Revoking access

The order does not matter.


New Access Token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. Learn more

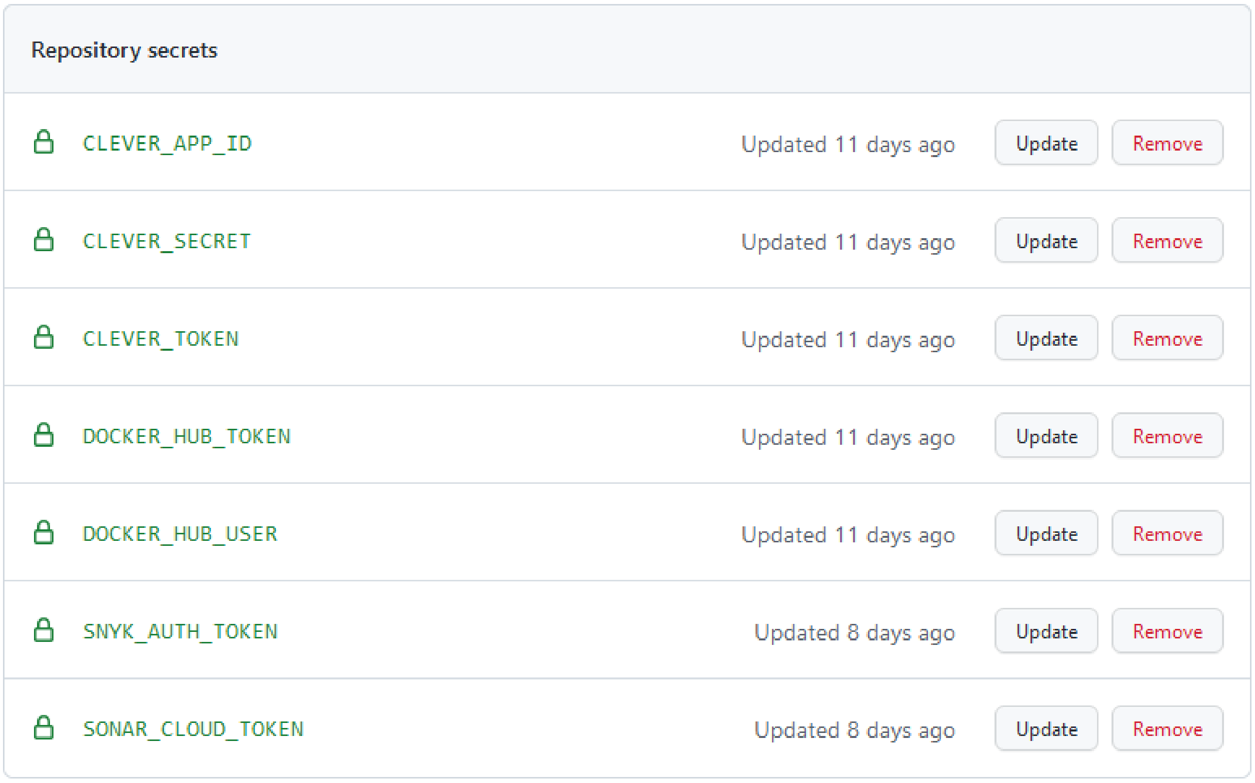
Access Token Description *
GitHub Action

Access permissions
Read, Write, Delete

Read, Write, Delete tokens allow you to manage your repositories.

New Access Token

To push to Docker Hub, we need to generate a token at https://hub.docker.com/settings/security and add it as a repository secret by the name `DOCKER_HUB_TOKEN` at https://github.com/Squoss/Fixadat/settings/secrets/actions.



Repository secrets

CLEVER_APP_ID	Updated 11 days ago	Update	Remove
CLEVER_SECRET	Updated 11 days ago	Update	Remove
CLEVER_TOKEN	Updated 11 days ago	Update	Remove
DOCKER_HUB_TOKEN	Updated 11 days ago	Update	Remove
DOCKER_HUB_USER	Updated 11 days ago	Update	Remove
SNYK_AUTH_TOKEN	Updated 8 days ago	Update	Remove
SONAR_CLOUD_TOKEN	Updated 8 days ago	Update	Remove

GitHub repository secrets

And even though it is not really a secret, we add the Docker Hub user name as a repository secret by the name `DOCKER_HUB_USER` at https://github.com/Squoss/Fixadat/settings/secrets/actions.

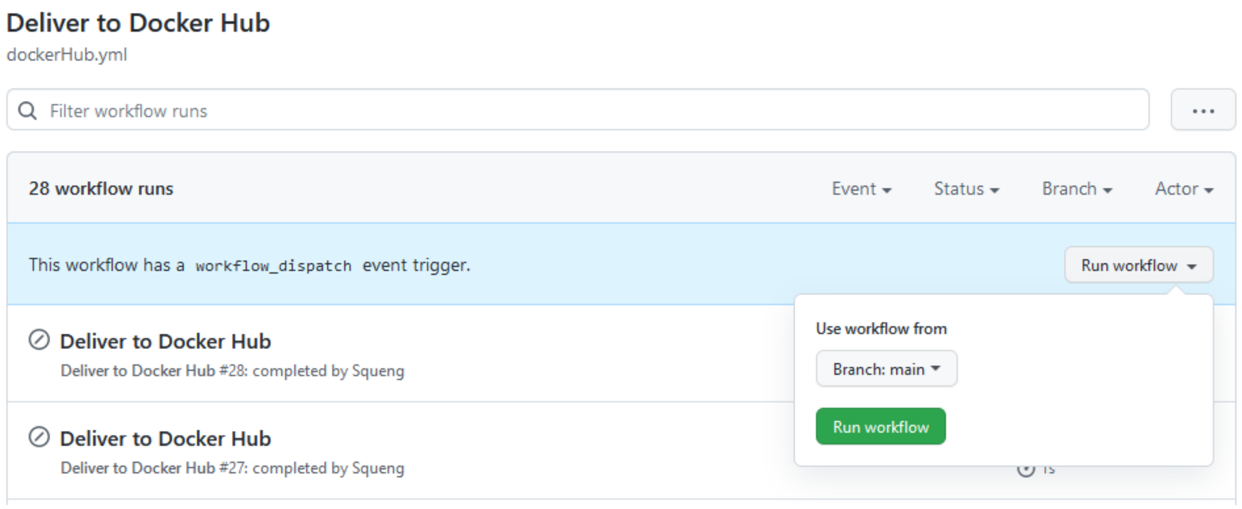
Now we can add the following as `dockerHub.yml`[331] in `Fixadat/.github/workflows`[332]:

```
name: Deliver to Docker Hub

on:
  workflow_run:
    workflows: ["Scan"]
    types: [completed]
  workflow_dispatch:

jobs:
  deliver:
    runs-on: ubuntu-latest
    if: ${{ github.event_name == 'workflow_dispatch' || github.event.workflow_run.co\
nclusion == 'success' }}
    steps:
```

[331]https://github.com/Squoss/Fixadat/blob/main/.github/workflows/dockerHub.yml
[332]https://github.com/Squoss/Fixadat/tree/main/.github/workflows

```
      - name: Set up Docker Buildx
        uses: docker/setup-buildx-action@v1
      - name: Log in to Docker Hub
        uses: docker/login-action@v1
        with:
          username: ${{ secrets.DOCKER_HUB_USER }}
          password: ${{ secrets.DOCKER_HUB_TOKEN }}
      - name: Build and push
        id: docker_build
        uses: docker/build-push-action@v2
        with:
          push: true
          tags: squeng/fixadat:latest
```

The latest image is built and pushed automatically whenever the scan from the Continuous Integration step passed. However, it could be the case that the scan fails because of a newly discovered vulnerability even though the image would be more secure than the currently delivered (i.e., published at Docker Hub) one. For such cases, we have inlcuded `workflow_dispatch` in order to build and push the latest image manually.


Deliver to Docker Hub
dockerHub.yml
28 workflow runs
This workflow has a `workflow_dispatch` event trigger.
Deliver to Docker Hub
Deliver to Docker Hub #28: completed by Squeng
Deliver to Docker Hub
Deliver to Docker Hub #27: completed by Squeng
Use workflow from
Branch: main

Manually deliver to Docker Hub

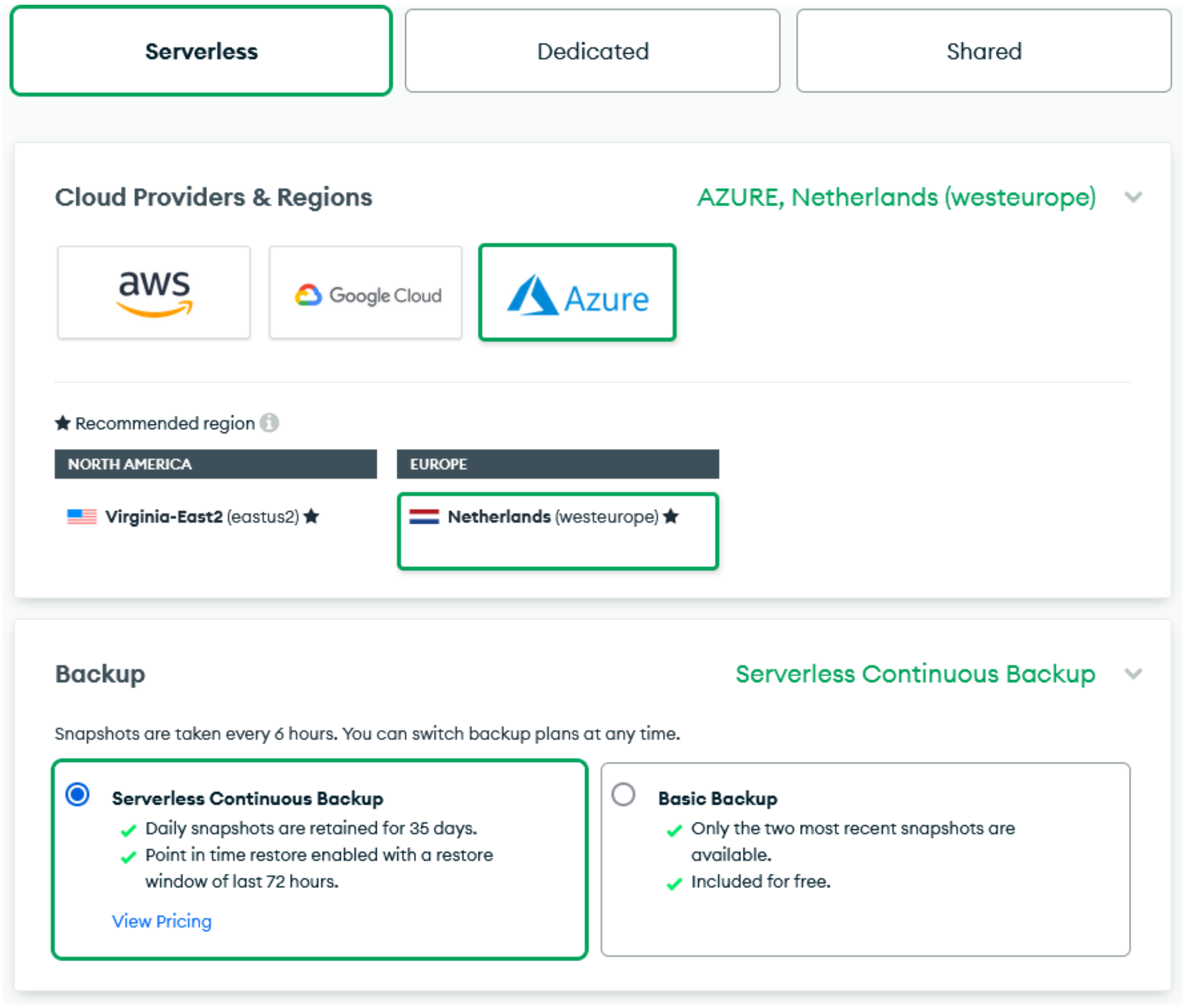
Continuous Deployment

At runtime, we need two environments, one for MongoDB and one for Docker. Nowadays, both are typically in the cloud.

MongoDB

There are quite a few MongoDB as a Service providers. One obvious choice would be to subscribe to the same provider as for the Platform as a Service (Clever Cloud[333] in our case). Another obvious choice is to subscribe to Atlas from MongoDB, Inc.[334].

[333]https://www.clever-cloud.com/mongodb-hosting/
[334]https://www.mongodb.com/atlas/database


Serverless
Dedicated
Shared

Cloud Providers & Regions — AZURE, Netherlands (westeurope)

★ Recommended region

NORTH AMERICA: Virginia-East2 (eastus2) ★
EUROPE: Netherlands (westeurope) ★

Backup — Serverless Continuous Backup

Snapshots are taken every 6 hours. You can switch backup plans at any time.

Serverless Continuous Backup
- ✓ Daily snapshots are retained for 35 days.
- ✓ Point in time restore enabled with a restore window of last 72 hours.

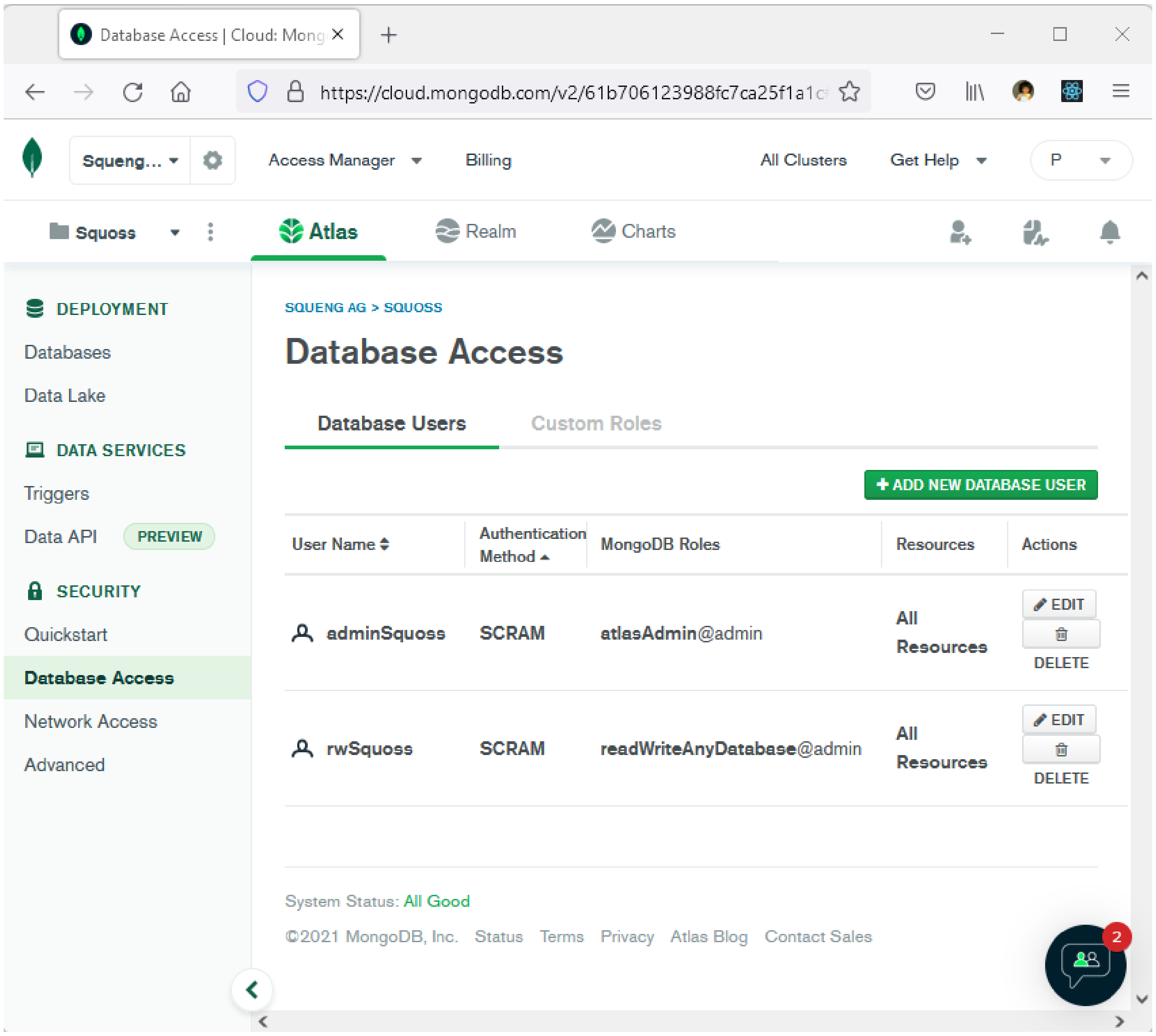
View Pricing

Basic Backup
- ✓ Only the two most recent snapshots are available.
- ✓ Included for free.

Atlas Create Serverless

Setting up an organization (e.g., "Squeng AG" in my case), then a project (e.g., "Squoss" in my case), and then a (free) cluster or a serverless instance with Atlas is straightforward[335].

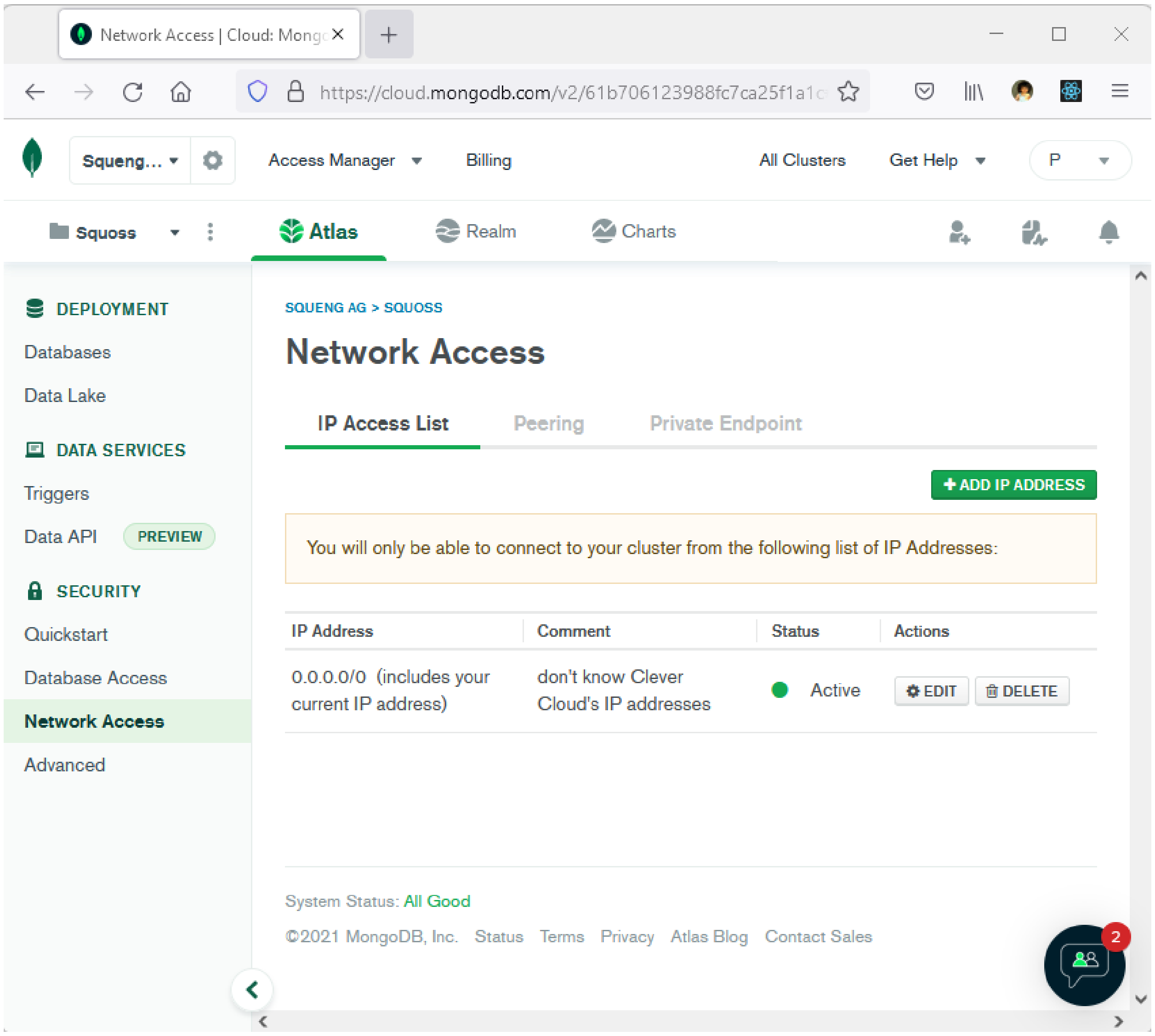
[335] https://docs.atlas.mongodb.com/getting-started/



User Name	Authentication Method	MongoDB Roles	Resources	Actions
adminSquoss	SCRAM	atlasAdmin@admin	All Resources	EDIT DELETE
rwSquoss	SCRAM	readWriteAnyDatabase@admin	All Resources	EDIT DELETE

Atlas Database Access

When it comes to database access, one should employ a user with read & write (but not more) access if possible. When it comes to network access, one should limit the IP addresses if possible.



You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
0.0.0.0/0 (includes your current IP address)	don't know Clever Cloud's IP addresses	● Active	EDIT DELETE

Atlas Network Access

In any case, a URI (e.g., `mongodb+srv://rwSquoss:PASSWORD@…?retryWrites=true&w=majority` in my case) should result as well as a DB name given or chosen (e.g., "fixadat" in my case).

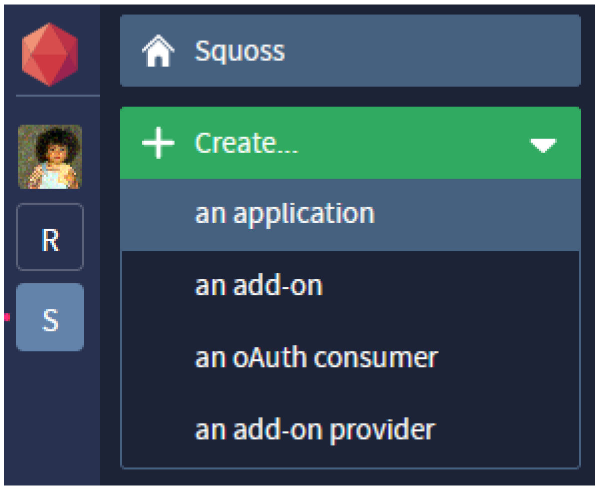
Docker

There are quite a few Platform as a Service providers that support Docker. I have not evaluated them in a while, but I have been a satisfied Clever Cloud[336] customer for years.

The Play documentation actually contains a page on Deploying to Clever Cloud[337], which you could read for the sake of completeness.

336 https://www.clever-cloud.com/
337 https://www.playframework.com/documentation/latest/Deploying-CleverCloud

Creating an account[338] and adding an organization[339] is straightforward. Applications can be created in one's personal space or in an organization.



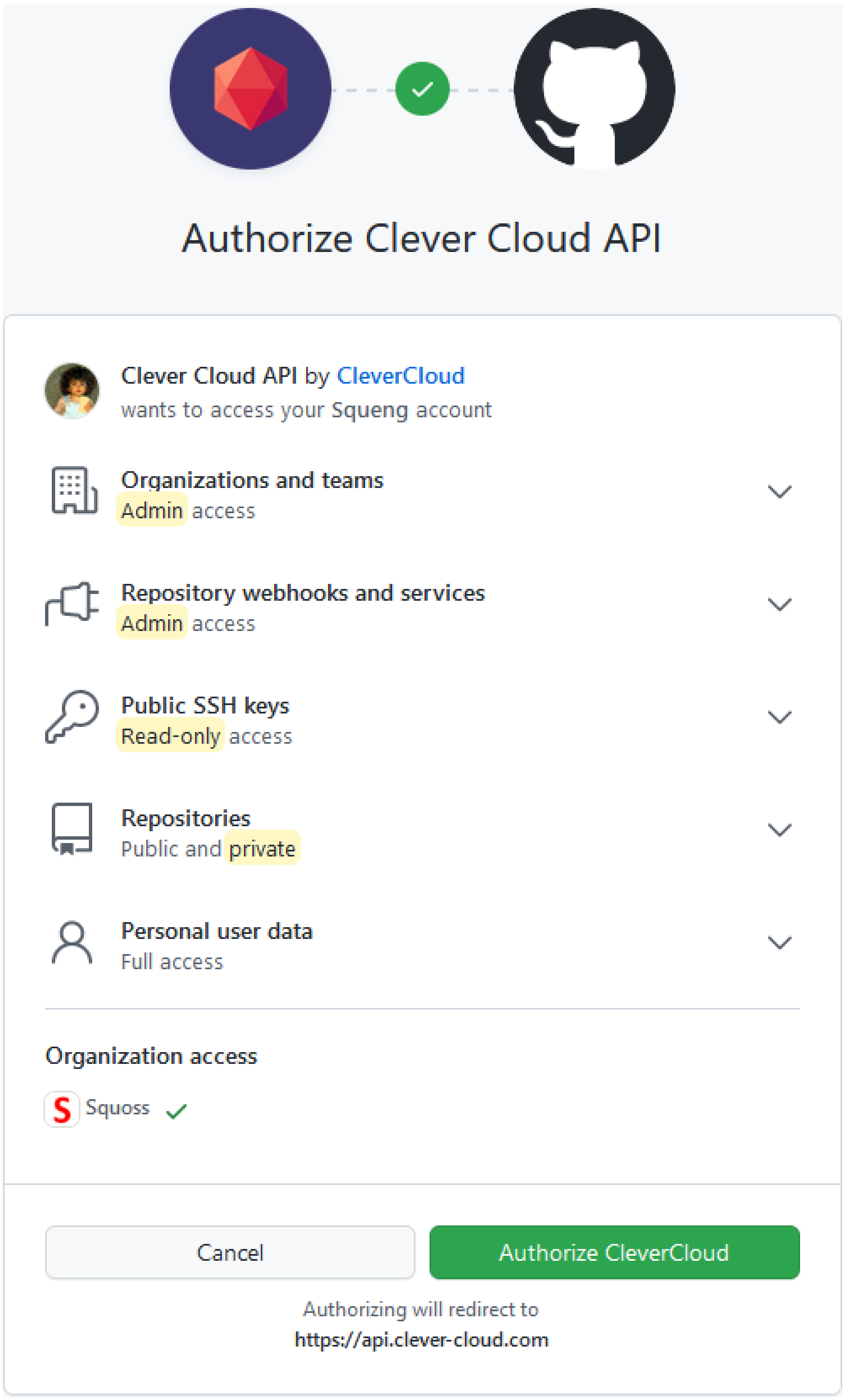
Create an application

Pull

If a project uses GitHub or Bitbucket, one approach for making the deployment continuous would be to link the Clever Cloud account to the GitHub or Bitbucket account via https://console.clever-cloud.com/users/me/information and then, when creating the application, to choose a source-code repository (and Docker as the application kind) in order for Clever Cloud to pull the sources and re-deploy the app automatically whenever they change. Such an approach would be relatively quick & dirty since it would not allow for any quality assurance by automated testing.

[338] https://www.clever-cloud.com/doc/account/create-account/
[339] https://www.clever-cloud.com/doc/account/administrate-organization/

⚠ Beware that Clever Cloud requests much more access to GitHub than it needs. That access could be revoked via https://github.com/settings/applications, but can unfortunately not be fine-tuned. So unless you deem Clever Cloud to be trustworthy, do not give it access to your GitHub account!



Authorize Clever Cloud API

Clever Cloud API by CleverCloud
wants to access your Squeng account

Organizations and teams
Admin access

Repository webhooks and services
Admin access

Public SSH keys
Read-only access

Repositories
Public and private

Personal user data
Full access

Organization access

Squoss ✓

Authorizing will redirect to
https://api.clever-cloud.com

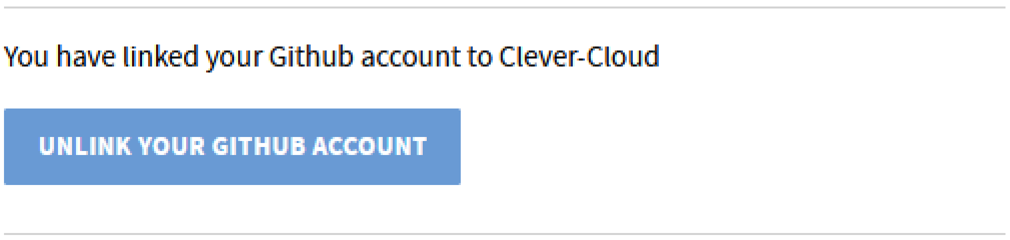
Linking Clever Cloud to GitHub

Push

The approach we are pursuing for making the deployment continuous is to extend the pipeline that we continued setting up in the last chapter. If you have given the pull approach above a try, you may want to clean it up & out first.

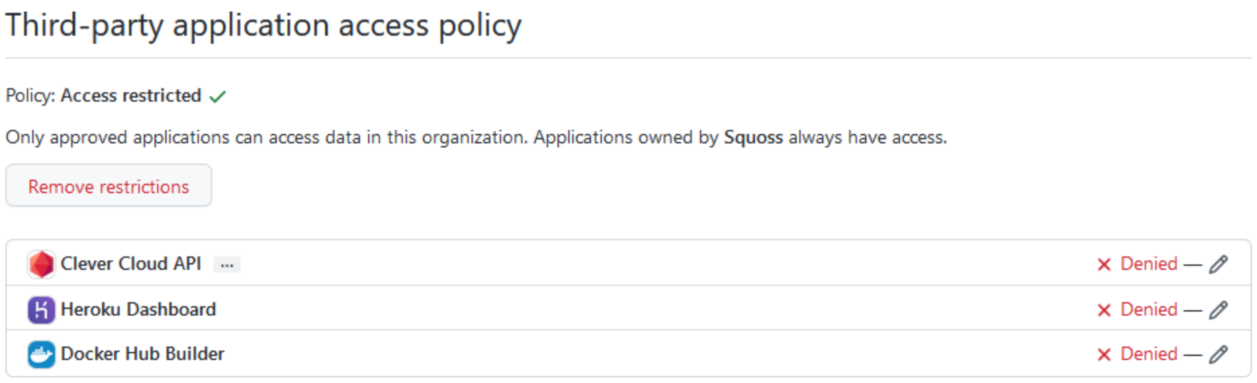
Cleaning up & out

If you have not already done so,


You have linked your Github account to Clever-Cloud
UNLINK YOUR GITHUB ACCOUNT

Unlinking accounts

- unlink your GitHub account from your Clever Cloud account at https://console.clever-cloud.com/users/me/information,
- deny Clever Cloud access to your GitHub organization (at https://github.com/organizations/Squoss/settings/oauth_application_policy in Fixadat's case), and
- revoke Clever Cloud's access to your personal account at https://github.com/settings/applications.



Third-party application access policy

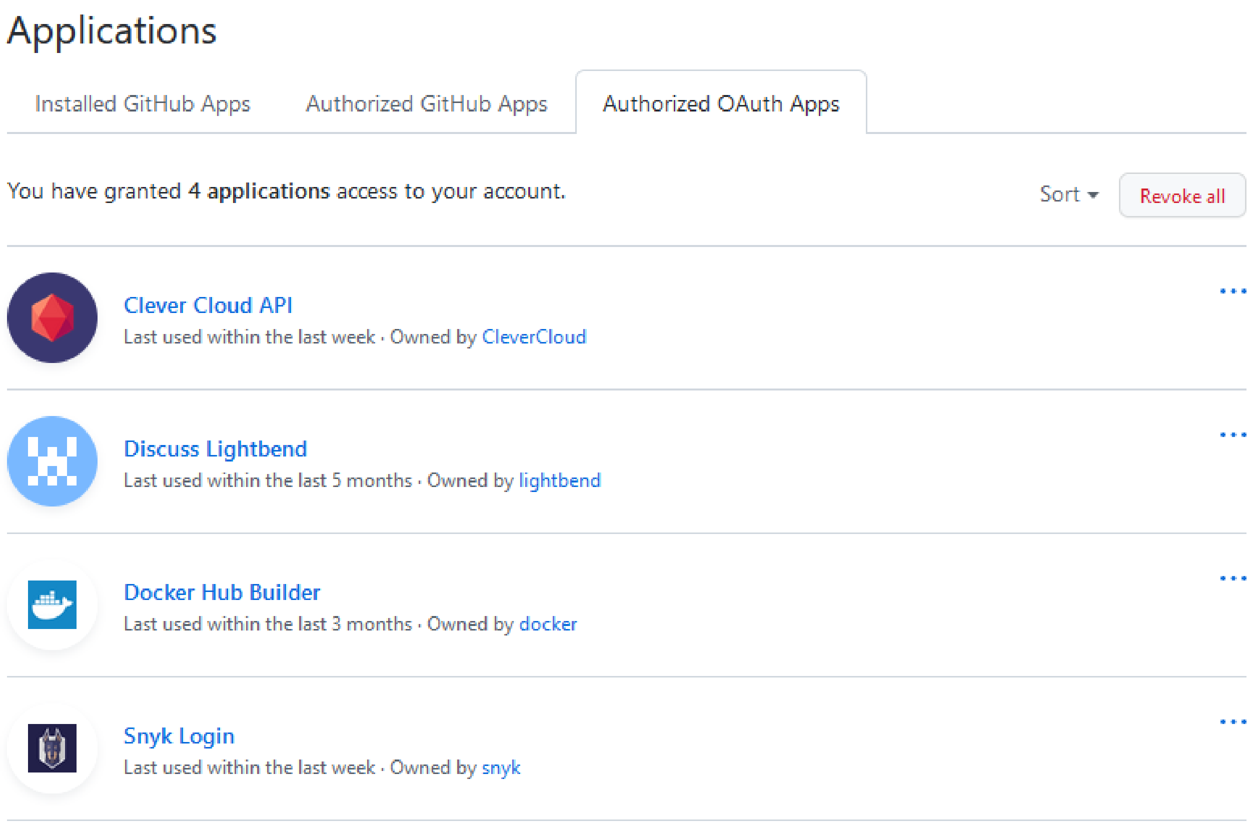
Policy: Access restricted ✓

Only approved applications can access data in this organization. Applications owned by **Squoss** always have access.

Clever Cloud API	✕ Denied —
Heroku Dashboard	✕ Denied —
Docker Hub Builder	✕ Denied —

Denying access

The order does not matter.



Applications

Installed GitHub Apps | Authorized GitHub Apps | Authorized OAuth Apps

You have granted **4 applications** access to your account.

Clever Cloud API
Last used within the last week · Owned by CleverCloud

Discuss Lightbend
Last used within the last 5 months · Owned by lightbend

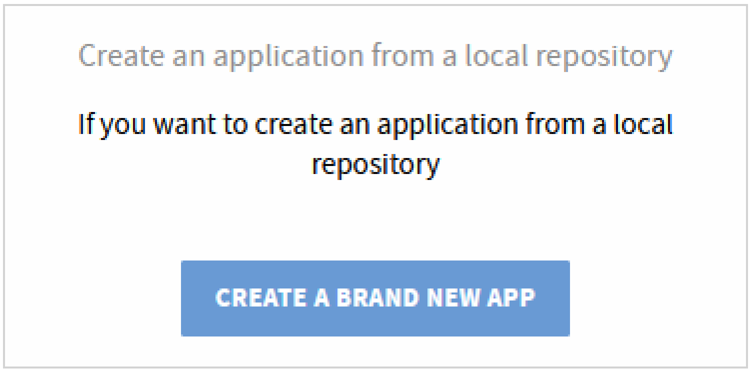
Docker Hub Builder
Last used within the last 3 months · Owned by docker

Snyk Login
Last used within the last week · Owned by snyk

Revoking access

Clever Cloud

Before we can push, we have to create an application by choosing to create a brand-new app.

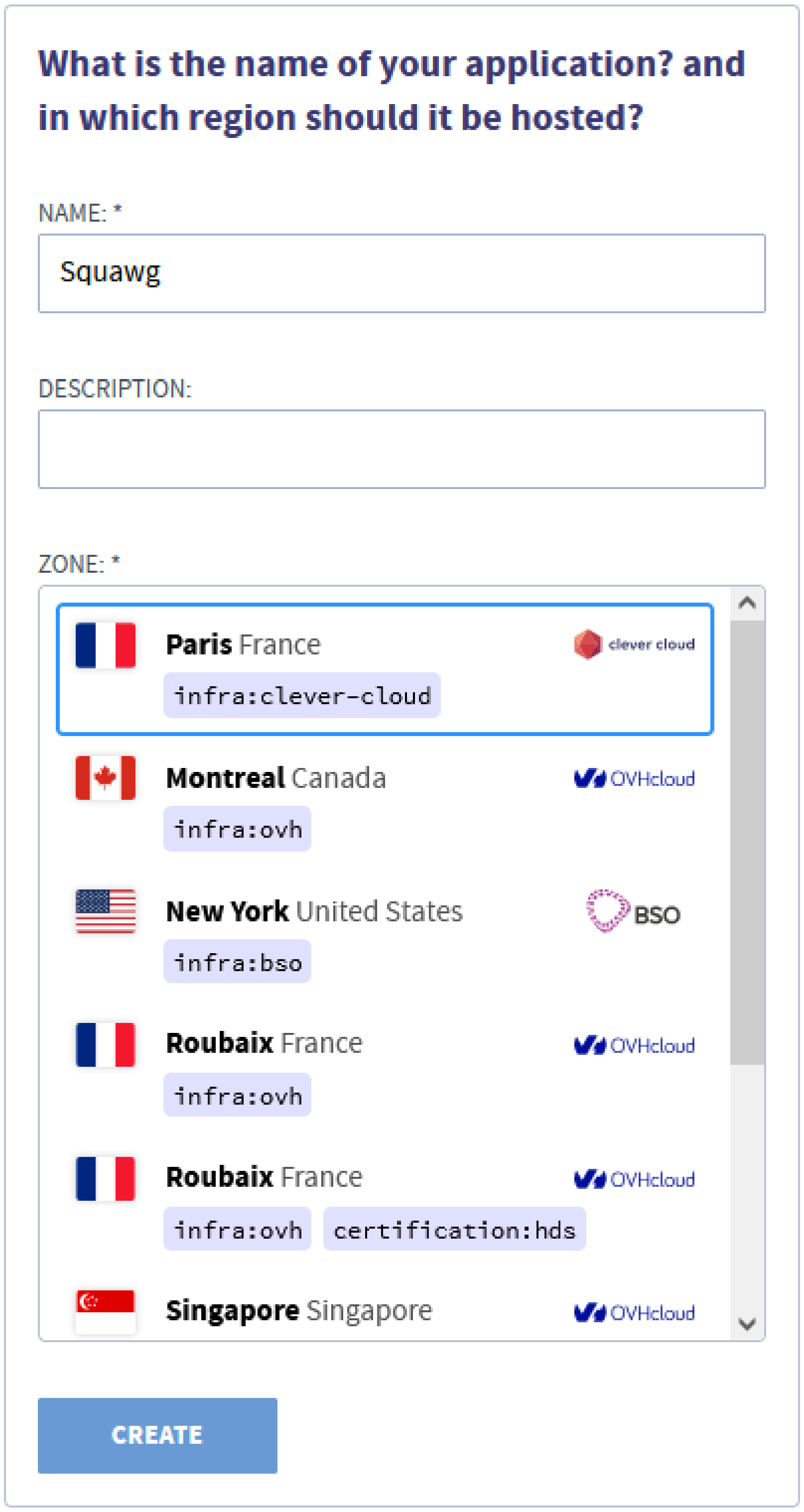


Create an application from a local repository

If you want to create an application from a local repository

Creating a brand-new app

We have to enter its name and choose a location of its cloud. (I have chosen Clever Cloud's own infrastructure in Paris; Fixadat running in Europe gives me a warm & fuzzy feeling with respect to data protection.)



What is the name of your application? and in which region should it be hosted?

NAME: *

Squawg

DESCRIPTION:

ZONE: *

Paris France — clever cloud
`infra:clever-cloud`

Montreal Canada — OVHcloud
`infra:ovh`

New York United States — BSO
`infra:bso`

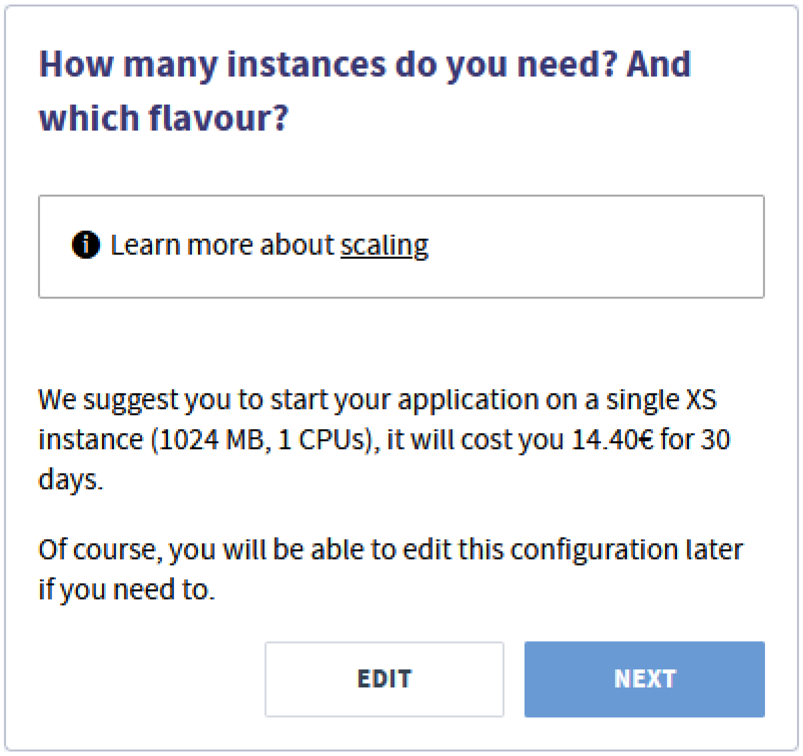
Roubaix France — OVHcloud
`infra:ovh`

Roubaix France — OVHcloud
`infra:ovh` `certification:hds`

Singapore Singapore — OVHcloud

Naming and locating the app

Then we can simply confirm the suggested dimensions.



How many instances do you need? And which flavour?

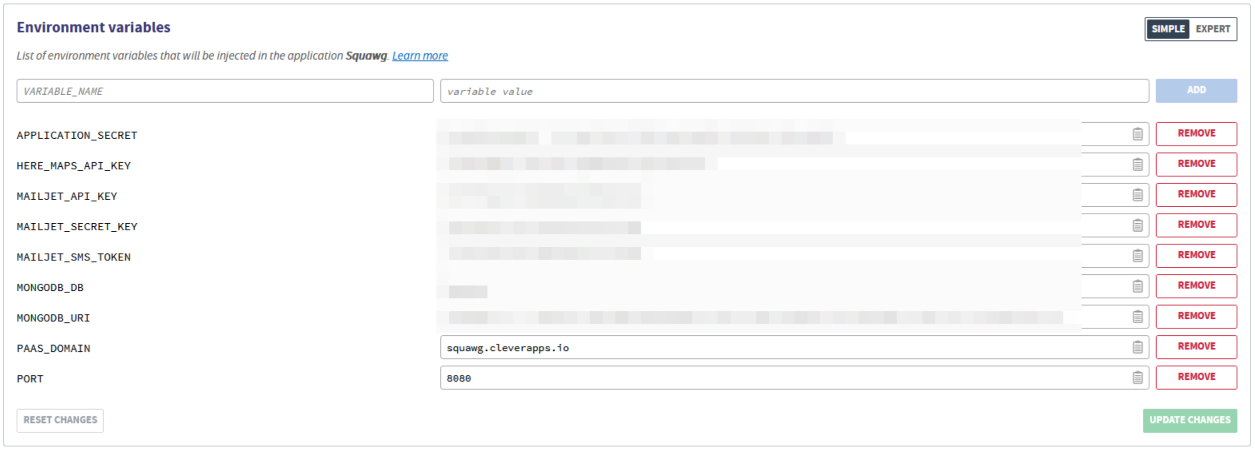
Learn more about scaling

We suggest you to start your application on a single XS instance (1024 MB, 1 CPUs), it will cost you 14.40€ for 30 days.

Of course, you will be able to edit this configuration later if you need to.

Starting small

Now would be the time to add MongoDB if one did not already have one. Then it is time to add all the environment variables referenced in `Fixadat/beapi/conf/application.conf`[340], including but not limited to the application secret. As you know[341], an application secret can be generated by executing `playGenerateSecret` in the sbt console.



Environment variables

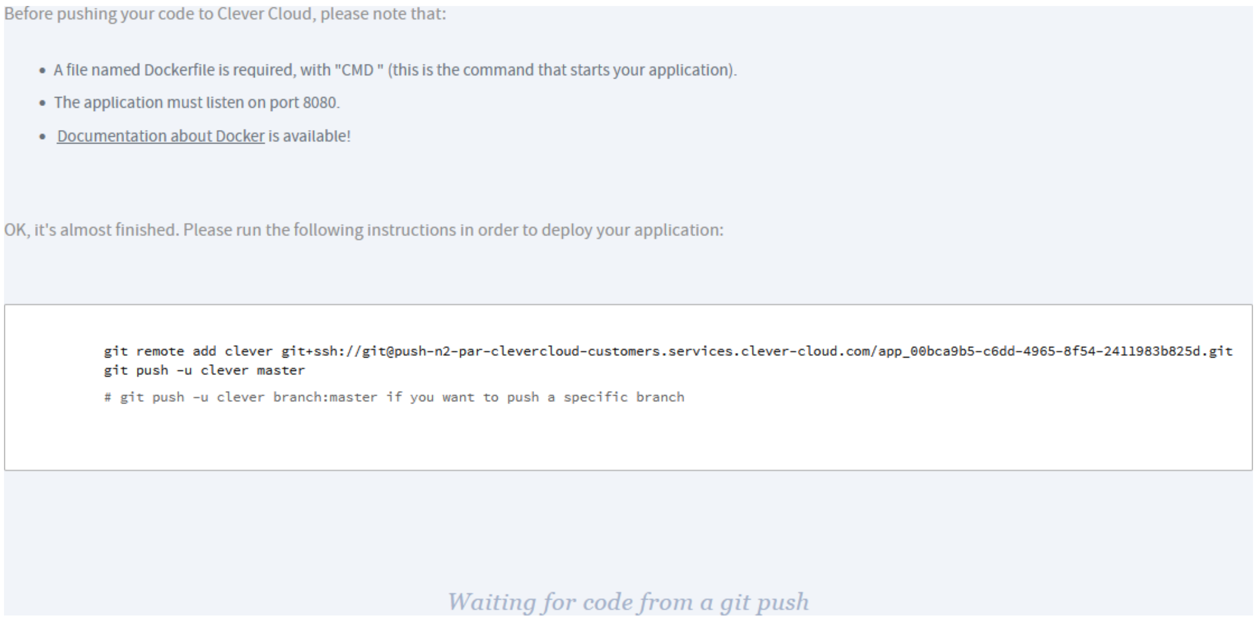
List of environment variables that will be injected in the application **Squawg**. Learn more

APPLICATION_SECRET	
HERE_MAPS_API_KEY	
MAILJET_API_KEY	
MAILJET_SECRET_KEY	
MAILJET_SMS_TOKEN	
MONGODB_DB	
MONGODB_URI	
PAAS_DOMAIN	squawg.cleverapps.io
PORT	8080

Environment variables

`PAAS_DOMAIN` will make sense once we reach the *Domains* section below.

[340]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/application.conf
[341]from https://www.playframework.com/documentation/latest/ApplicationSecret#Generating-an-application-secret



Before pushing your code to Clever Cloud, please note that:

- A file named Dockerfile is required, with "CMD " (this is the command that starts your application).
- The application must listen on port 8080.
- Documentation about Docker is available!

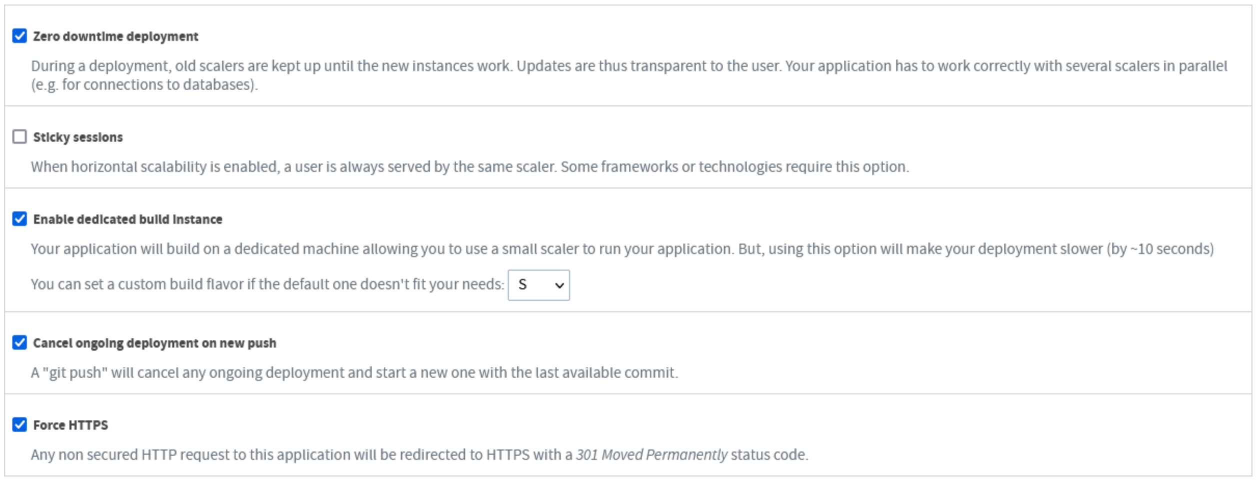
OK, it's almost finished. Please run the following instructions in order to deploy your application:

```
git remote add clever git+ssh://git@push-n2-par-clevercloud-customers.services.clever-cloud.com/app_00bca9b5-c6dd-4965-8f54-2411983b825d.git
git push -u clever master
# git push -u clever branch:master if you want to push a specific branch
```

Waiting for code from a git push

Waiting

While Clever Cloud waits for the first deployment, we can provide it with further pieces of information. On Clever Cloud's end, there is more to configure. First, because XS is too small to build the app, we have to enable a bigger, dedicated build instance. Second, as announced, we should force HTTPS. Third, we want user-friendlier (and marketable) domains.



[✓] **Zero downtime deployment**
During a deployment, old scalers are kept up until the new instances work. Updates are thus transparent to the user. Your application has to work correctly with several scalers in parallel (e.g. for connections to databases).

[ ] **Sticky sessions**
When horizontal scalability is enabled, a user is always served by the same scaler. Some frameworks or technologies require this option.

[✓] **Enable dedicated build instance**
Your application will build on a dedicated machine allowing you to use a small scaler to run your application. But, using this option will make your deployment slower (by ~10 seconds)
You can set a custom build flavor if the default one doesn't fit your needs: S

[✓] **Cancel ongoing deployment on new push**
A "git push" will cancel any ongoing deployment and start a new one with the last available commit.

[✓] **Force HTTPS**
Any non secured HTTP request to this application will be redirected to HTTPS with a *301 Moved Permanently* status code.

Information

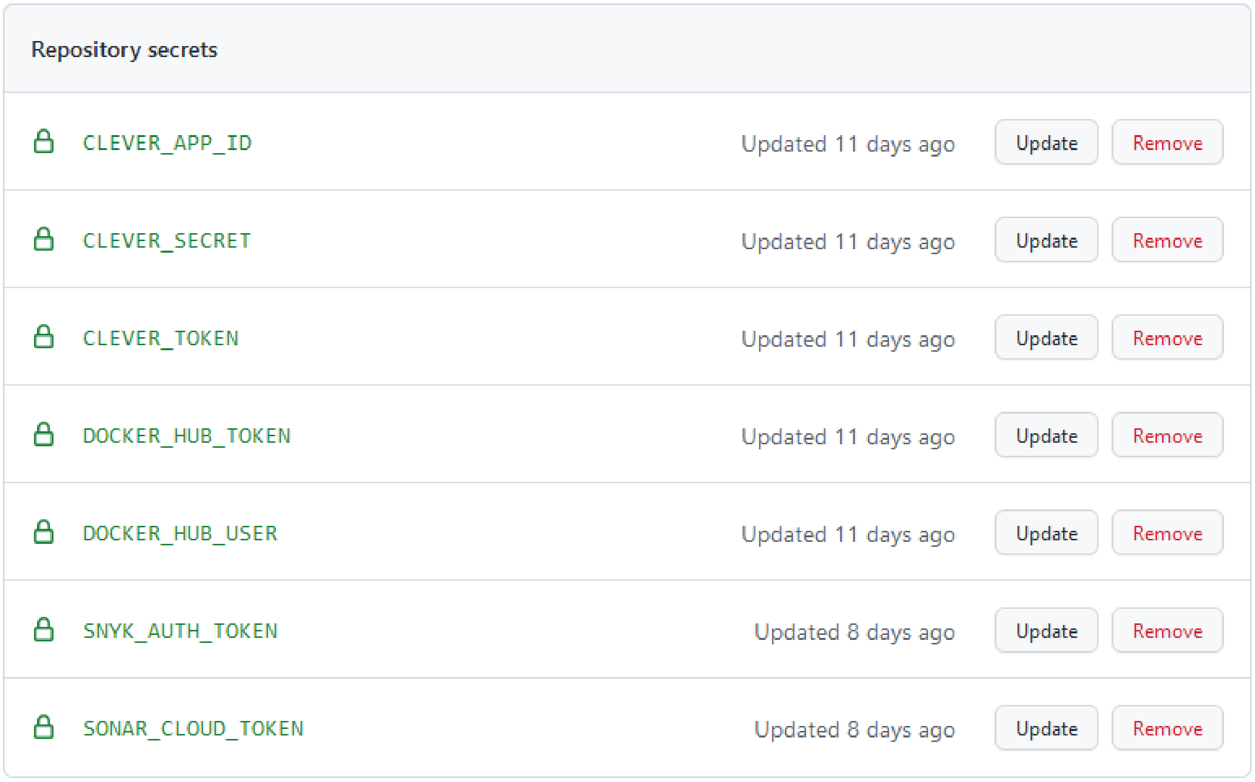
FIXME/TODO: In the context of Clever Cloud, Fixadat has to trust all proxies[342] and use the legacy X-Forwarded headers[343]. That's because the TLS connection terminates at the edge (i.e., upon connecting to Clever Cloud) and is then forwarded without TLS. We could configure the app accordingly by adding `play.http.forwarded.trustedProxies`

[342]https://www.playframework.com/documentation/latest/HTTPServer#Trusting-all-proxies
[343]https://www.playframework.com/documentation/latest/HTTPServer#Forwarded-header-version

= ["0.0.0.0/0", "::/0"] and play.http.forwarded.version = "x-forwarded" to Fixadat/beapi/conf/application.conf[344]. However, that would result in Clever-Cloud-specific configuration in the general configuration. Alternatively, we could have a https://www.playframework.com/documentation/latest/ProductionConfiguration x-forwarded is the default (play.http.forwarded.version,Quoted("x-forwarded"), play.filters.https.xForwardedProtoEnabled,Config
but so is ["127.0.0.1","::1"] ((play.http.forwarded.trustedProxies,SimpleConfigList(["127.0.0.1","::1"])))
AND YET IT ALREADY WORKS?! tbd

GitHub

To push to Clever Cloud, we need to generate a token as well as a secret as explained by 47ng[345] and add them as repository secrets by the name CLEVER_TOKEN and CLEVER_SECRET, respectively, at https://github.com/Squoss/Fixadat/settings/secrets/actions.


Repository secrets

CLEVER_APP_ID	Updated 11 days ago	Update	Remove
CLEVER_SECRET	Updated 11 days ago	Update	Remove
CLEVER_TOKEN	Updated 11 days ago	Update	Remove
DOCKER_HUB_TOKEN	Updated 11 days ago	Update	Remove
DOCKER_HUB_USER	Updated 11 days ago	Update	Remove
SNYK_AUTH_TOKEN	Updated 8 days ago	Update	Remove
SONAR_CLOUD_TOKEN	Updated 8 days ago	Update	Remove

GitHub repository secrets

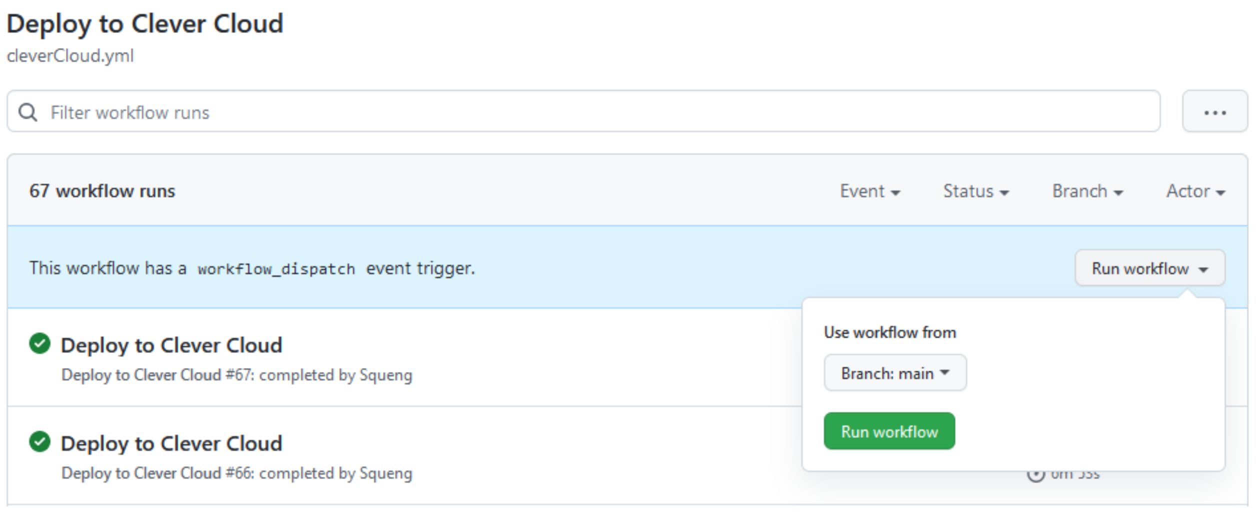
And even though it is not really a secret, we add the Clever Cloud app ID as a repository secret by the name CLEVER_APP_ID at https://github.com/Squoss/Fixadat/settings/secrets/actions.

Now we can add the following as cleverCloud.yml[346] in Fixadat/.github/workflows[347]:

[344]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/application.conf
[345]https://github.com/47ng/actions-clever-cloud#authentication
[346]https://github.com/Squoss/Fixadat/blob/main/.github/workflows/cleverCloud.yml
[347]https://github.com/Squoss/Fixadat/tree/main/.github/workflows

```yaml
name: Deploy to Clever Cloud

on:
  workflow_run:
    workflows: ["Scan"]
    types: [completed]
  workflow_dispatch:

jobs:
  deploy:
    runs-on: ubuntu-latest
    if: ${{ github.event_name == 'workflow_dispatch' || github.event.workflow_run.co\
nclusion == 'success' }}
    steps:
    - uses: actions/checkout@v2
      with:
        fetch-depth: 0
    - name: Set up Node.js 16
      uses: actions/setup-node@v2
      with:
        node-version: '16'
    - name: Install the Clever Tools # https://www.clever-cloud.com/doc/reference/cl\
ever-tools/getting_started/#via-npm
      run: npm install -g clever-tools
    - name: Link to the Clever Cloud account # https://www.clever-cloud.com/doc/refe\
rence/clever-tools/getting_started/#linking-your-account
      run: clever login --token ${{ secrets.CLEVER_TOKEN }} --secret ${{ secrets.CLE\
VER_SECRET }}
    - name: Link to the app # https://www.clever-cloud.com/doc/reference/clever-tool\
s/getting_started/#linking-an-existing-application
      run: clever link ${{ secrets.CLEVER_APP_ID }}
    - name: Deploy # https://www.clever-cloud.com/doc/reference/clever-tools/getting\
_started/#deploying-new-code
      run: clever deploy
```

The latest repository is pushed automatically whenever the scan from the Continuous Integration step passed. However, it could be the case that the scan fails because of a newly discovered vulnerability even though the resulting container would be more secure than the currently deployed one. For such cases, we have inlcuded `workflow_dispatch` in order to push the latest repository manually.



Deploy to Clever Cloud
cleverCloud.yml

Filter workflow runs

67 workflow runs — Event, Status, Branch, Actor

This workflow has a `workflow_dispatch` event trigger. Run workflow

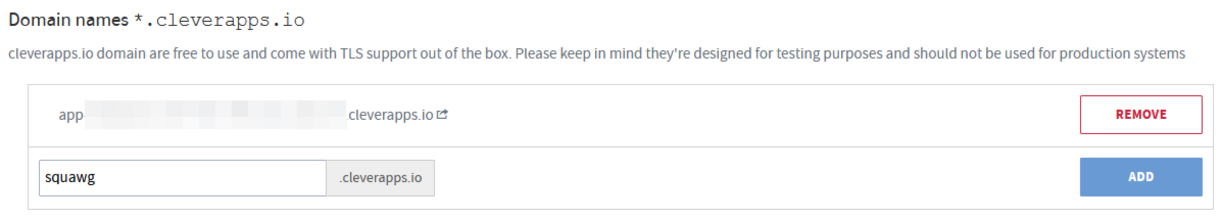
Use workflow from
Branch: main
Run workflow

Deploy to Clever Cloud
Deploy to Clever Cloud #67: completed by Squeng

Deploy to Clever Cloud
Deploy to Clever Cloud #66: completed by Squeng

Manually deploy to Clever Cloud

Domains



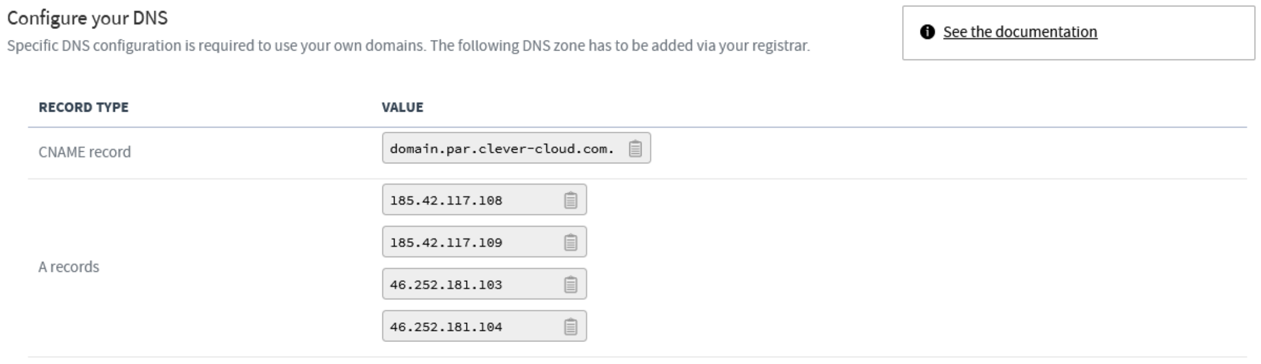
Domain names `*.cleverapps.io`

cleverapps.io domain are free to use and come with TLS support out of the box. Please keep in mind they're designed for testing purposes and should not be used for production systems

app[redacted]cleverapps.io	REMOVE
squawg .cleverapps.io	ADD

cleverapps.io domain

A user-friendlier (but not marketable) domain can easily be added. As shown above, it also needs to be allow-listed via the environment variable `PAAS_DOMAIN`. However, we would like to use the top-level domain that was already allow-listed in `Fixadat/beapi/conf/application.conf`[348].



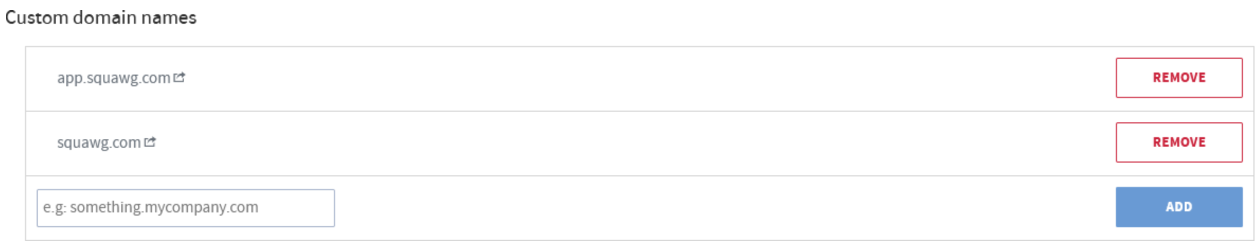
Configure your DNS
Specific DNS configuration is required to use your own domains. The following DNS zone has to be added via your registrar.

See the documentation

RECORD TYPE	VALUE
CNAME record	`domain.par.clever-cloud.com.`
A records	`185.42.117.108`
	`185.42.117.109`
	`46.252.181.103`
	`46.252.181.104`

App-specific values

[348]https://github.com/Squoss/Fixadat/blob/main/beapi/conf/application.conf


Custom domain names

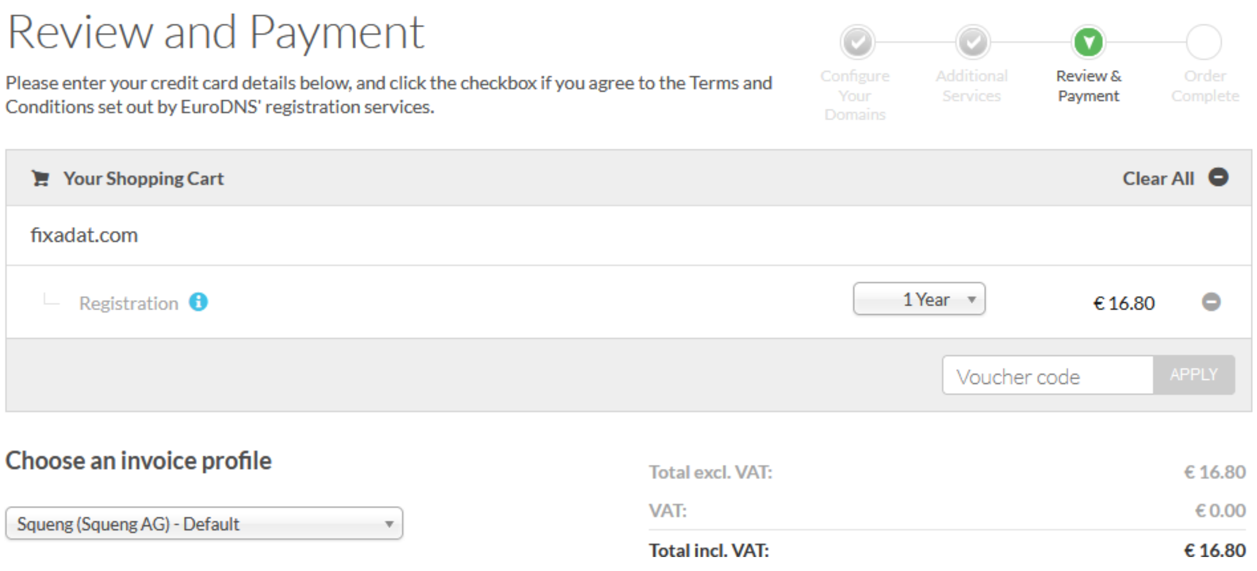
app.squawg.com REMOVE

squawg.com REMOVE

e.g: something.mycompany.com ADD

Top-level domain

With the help of a contextual example[349] and the app (in our case Paris)-specific values, managing the DNS zone is straightforward.


Review and Payment

Please enter your credit card details below, and click the checkbox if you agree to the Terms and Conditions set out by EuroDNS' registration services.

Your Shopping Cart — Clear All

fixadat.com

Registration	1 Year	€ 16.80	

Voucher code APPLY

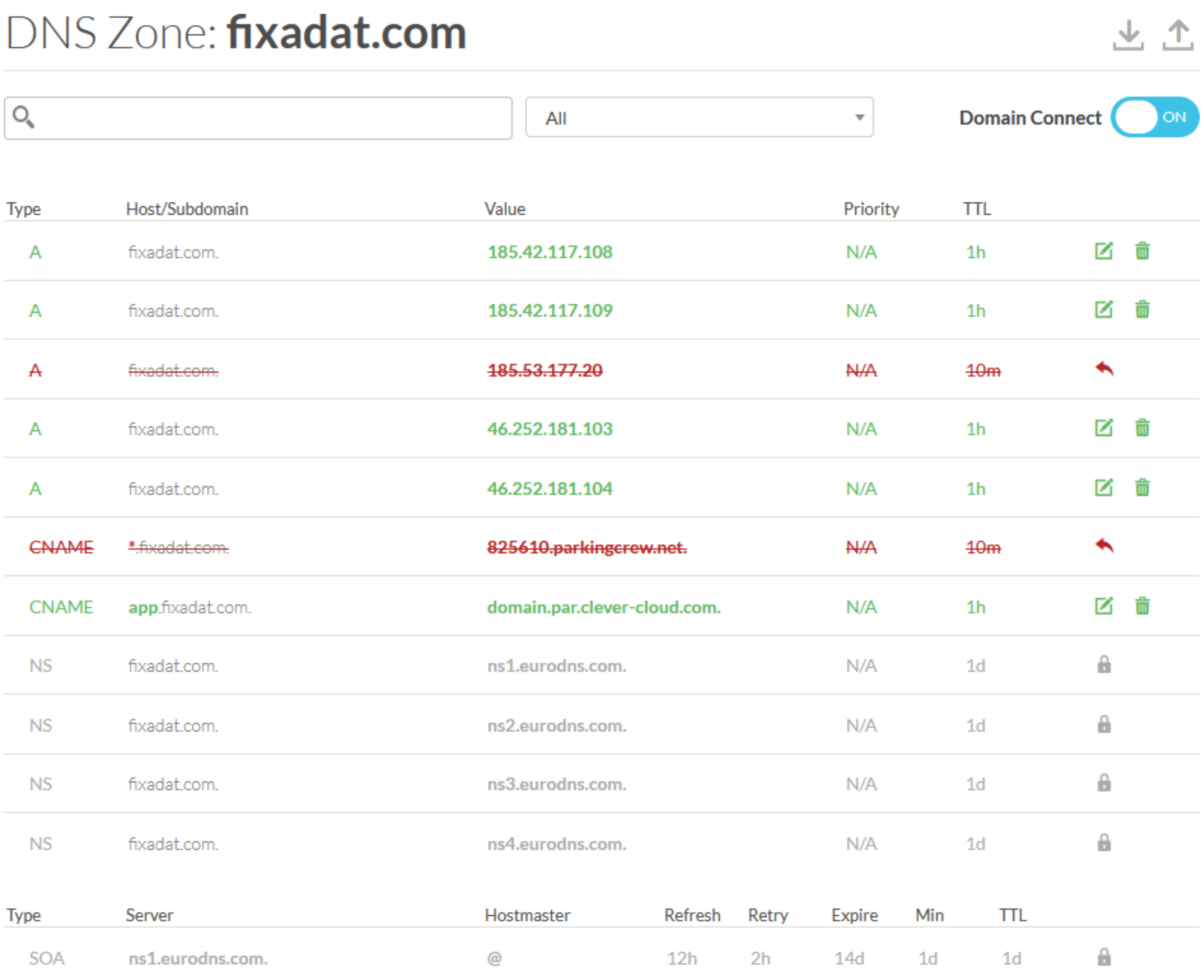
Choose an invoice profile

Squeng (Squeng AG) - Default

Total excl. VAT:	€ 16.80
VAT:	€ 0.00
Total incl. VAT:	€ 16.80

Ordering fixadat.com

[349] https://www.clever-cloud.com/doc/administrate/domain-names/#contextual-example



DNS Zone: **fixadat.com**

All

Domain Connect ON

Type	Host/Subdomain	Value	Priority	TTL
A	fixadat.com.	185.42.117.108	N/A	1h
A	fixadat.com.	185.42.117.109	N/A	1h
~~A~~	~~fixadat.com.~~	~~185.53.177.20~~	~~N/A~~	~~10m~~
A	fixadat.com.	46.252.181.103	N/A	1h
A	fixadat.com.	46.252.181.104	N/A	1h
~~CNAME~~	~~*.fixadat.com.~~	~~825610.parkingcrew.net.~~	~~N/A~~	~~10m~~
CNAME	app.fixadat.com.	domain.par.clever-cloud.com.	N/A	1h
NS	fixadat.com.	ns1.eurodns.com.	N/A	1d
NS	fixadat.com.	ns2.eurodns.com.	N/A	1d
NS	fixadat.com.	ns3.eurodns.com.	N/A	1d
NS	fixadat.com.	ns4.eurodns.com.	N/A	1d

Type	Server	Hostmaster	Refresh	Retry	Expire	Min	TTL
SOA	ns1.eurodns.com.	@	12h	2h	14d	1d	1d

Managing fixadat.com

Appendix

Updating and Upgrading

At the very least for security reasons, you need to update and eventually upgrade your projects on a regular basis.

Updating the Back-end

- update `FROM hseeberger/scala-sbt:JAVA_SBT_SCALA as play` in `Fixadat/Dockerfile`
- follow the migration instructions[350] if need be
- update `scalaVersion` in `Fixadat/beapi/build.sbt` as well as in `Fixadat/beapi/reinraum/build.sbt`
- update `sbt.version` in `Fixadat/beapi/project/build.properties` as well as in `Fixadat/beapi/reinraum/project/build.properties`
- update `addSbtPlugin("com.typesafe.play" % "sbt-plugin" % "X.Y.Z")` in `Fixadat/beapi/project/plugins.sbt`
- update third-party dependencies in both `Fixadat/beapi/build.sbt` and `Fixadat/beapi/reinraum/build.sbt`

Updating the Front-end

- follow the migration instructions[351] if need be
- with your shell, execute the command `npm update` in `Fixadat/fegui`
- with your shell, execute the command `npm audit fix` in `Fixadat/fegui`

Upgrading the Back-end

- upgrade `FROM openjdk:JAVA` as well as `FROM hseeberger/scala-sbt:JAVA_SBT_SCALA as play` in `Fixadat/Dockerfile`
- follow the migration instructions[352]
- to be continued

Upgrading the Front-end

- upgrade `FROM node:JS as react` in `Fixadat/Dockerfile`
- follow the migration instructions[353]
- to be continued

[350] https://www.playframework.com/documentation/latest/Migration28
[351] https://github.com/facebook/create-react-app/blob/main/CHANGELOG.md
[352] https://www.playframework.com/documentation/latest/Migration28
[353] https://github.com/facebook/create-react-app/blob/main/CHANGELOG.md