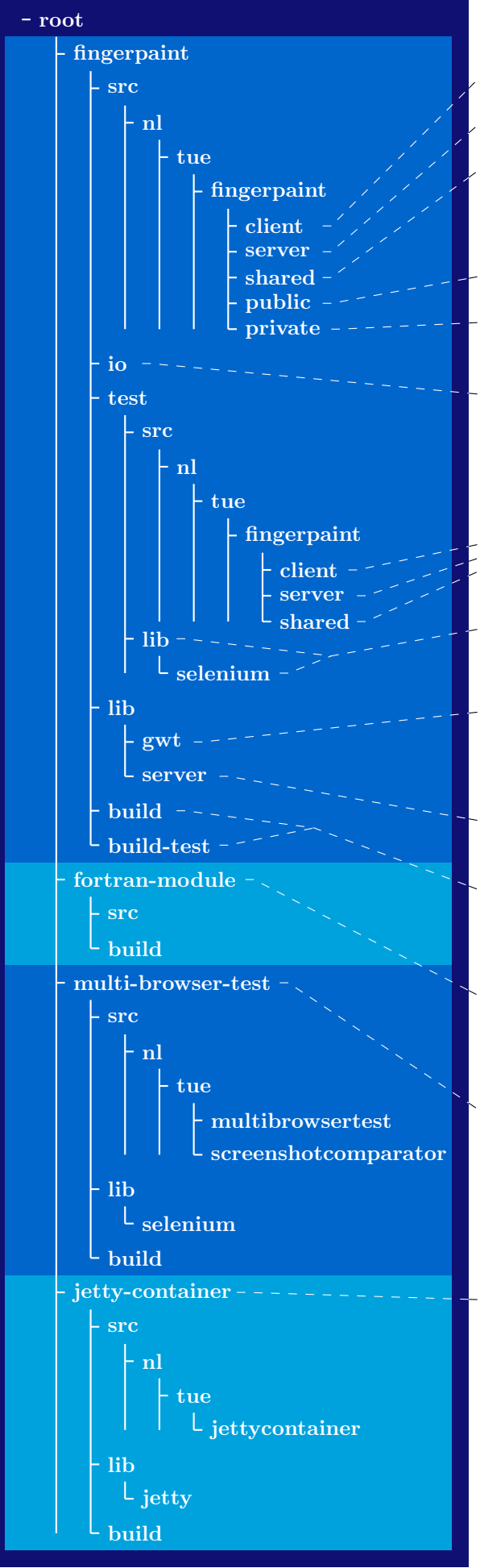

The `client` package contains all code that is compiled into JavaScript by GWT.

The `server` package contains all code that is compiled into Java bytecode and run on the server.

The `shared` package contains all code that is both used in the `client` and the `sever` package. Thus, it will be both compiled into the JavaScript that is run on the client-side and the Java bytecode that is run on the server-side.

The `public` package contains all publicly available static files, such as HTML and images.

The `private` package contains all non publicly available static files. For example, the mapping matrices are stored here.

The `io` package contains a library that is used to make buttons respond faster on the iPad. It should be moved to a library folder, but we did not have time for that.

These packages contain tests for the classes in the same packages in the `src` folder.

We use Selenium for automated testing. All libraries used only for testing are put here.

The `gwt` package contains libraries that GWT uses to compile Java into JavaScript and also contains libraries that are used on the client side and thus need to be compiled *by GWT* into JavaScript.

The `server` package contains libraries that are used on the server-side. Note that these may be `jar`-files or even shared libraries (`dll`- or `so`-files).

The `build` folder contains temporary files that are generated during compilation. The same holds for the `build-test` folder, but that one is used when compiling tests. Both are auto-generated.

The **fortran-module** is the piece of code that actually performs simulations. The FINGERPAINT application depends on this module and communicates with it through JNI.

The **multi-browser-test** project provides ways to easily run automated tests in multiple browsers (through Selenium) and take and compare screenshots. We developed this for the project and use this in **fingerpaint** for testing purposes.

The **jetty-container** project enables us to package **fingerpaint** together with Jetty in an executable `jar`-file. This can be used to deploy and run the FINGERPAINT application without having to install an instance of Jetty on a machine.