Database Development

a) Command Line Programs

a.i. Haskell Database Development

Haskell supports a variety of SQL Databases including Mysql, Sqlite and MariaDB. I've already developed a database migration tool using haskell together with MySql. This migration tool was able to perform several db functions such as creating or deleting a user, inserting an API key, importing a csv file into the database, etc. The options of the database tool were defined by implementing a complex datatype.

```
data MigrationOpts
  = CreateDB
  | CreateUser { userName :: Text
               , userPassword :: Password
               , privateDefault :: Maybe Bool
               , archiveDefault :: Maybe Bool
               , privacyLock :: Maybe Bool }
  | CreateApiKey { userName :: Text }
  | DeleteApiKey { userName :: Text }
  | DeleteUser { userName :: Text }
  | ImportBookmarks { userName :: Text
                    , bookmarkFile :: FilePath }
  | ImportMyBookmarks { userName :: Text
                      , bookmarkFile :: FilePath }
  | ImportFirefoxBookmarks { userName :: Text
                           , bookmarkFile :: FilePath }
  | ExportBookmarks { userName :: Text
                    , bookmarkFile :: FilePath }
  | ImportNotes { userName :: Text
                , noteDirectory :: FilePath }
  | PrintMigrateDB
  deriving (Generic, Show)
```

Figure i: Command line options of a database tool

The db tool was developed in such a way so the user didn't have to enter the user credentials each time he or she was accessing the database. That's why I stored the data used for connecting to the database in a textfile using key value pairs.

```
# database configuration
[MYSQL]
user = dbuser
password = dbpw
host = localhost
port = 3306
database = mydb
# Options: For example: "opt1,opt2,opt3"
options =
config = config/queries.sql
```

Figure ii: Configuration file used for storing the database settings

a.ii. Java command line tools

I can realize a database tool in Java using a text based user interface (**TUI**). This allows for more intuition when using this tool as no command is entered directly on the console. Under Java I prefer the Sql library **Jooq** over the others as this library offers a wide range of sql functions and its syntax is highly readable which makes it easier to use.

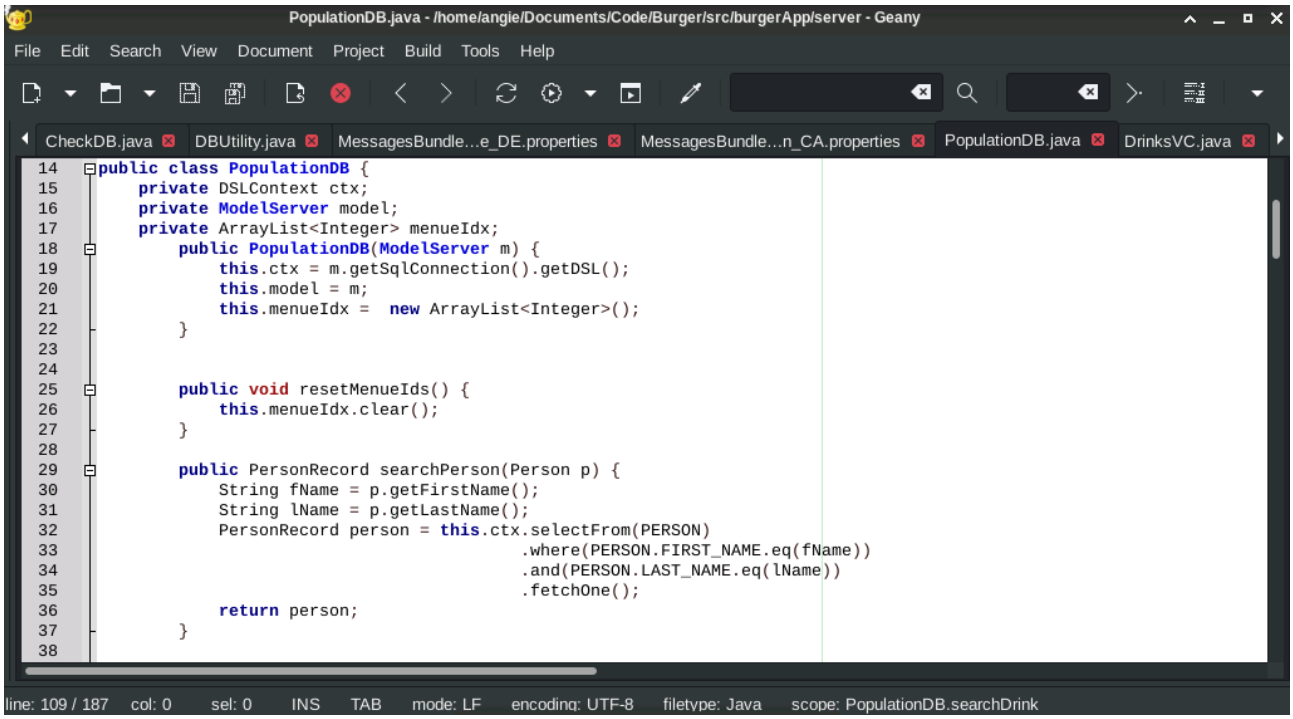

Figure iii: Using Jooq, sql queries are written in pure Java syntax

I've implemented several database programs using Java and **lanterna**. Lanterna is a library which is used for realizing a tui. Using a TUI allows for a clear user interface that is easy to use. Lanterna offers multiple TUI components such as tables, buttons, dropdown menues, etc.

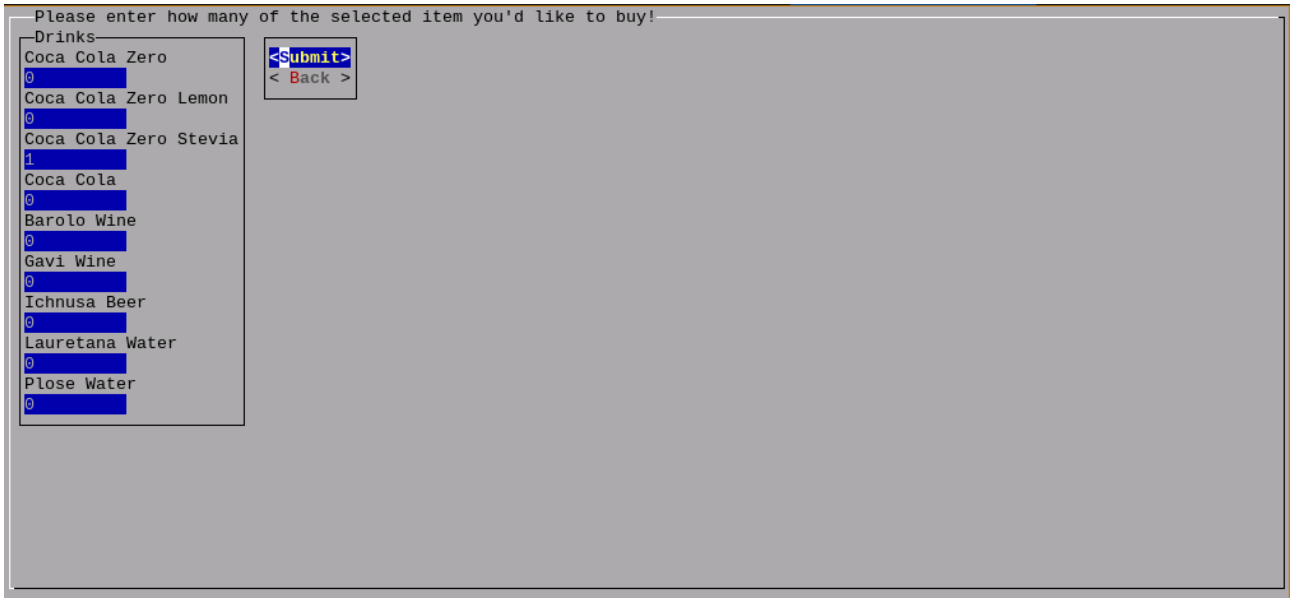

Figure iv: Textfields in a TUI realized with Lanterna

What's more, I can define events fired by pressing a button in Lanterna and write handlers for the events. This approach allows for a structured code so the UI elements will be realized in one class while retrieving the data from a database will take place in another class.

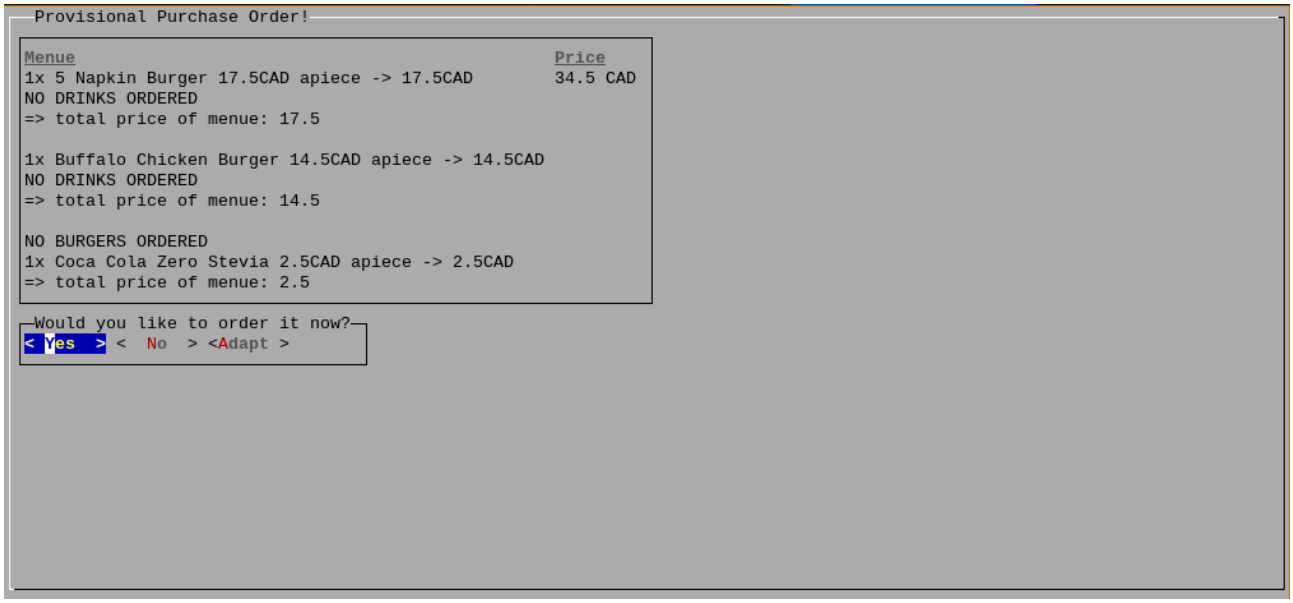

Figure v: The next event will take place as soon as a button is being pressed.

b) Web Applications / GUIs

b.i. JavaFX

Using the gui framework JavaFX makes it possible to implement GUIs with a liquid layout and a css styled user interface. I developed my own database application using java, jooq and JavaFX as the frontend library. MyDbApp offers a menu for adjusting the database settings such as username, password, etc. The usage of MyDbApp is password protected which allows for more security.

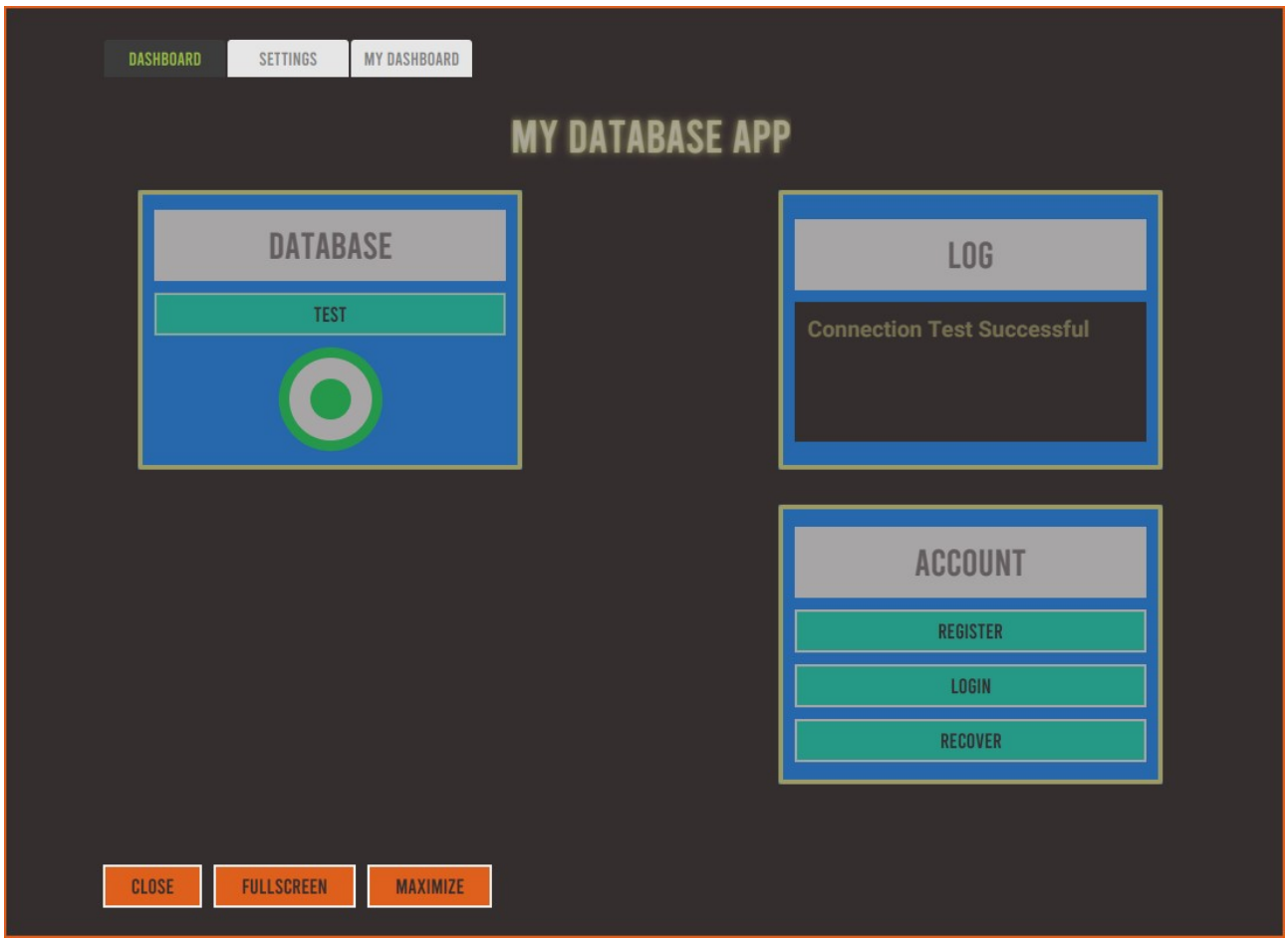

Figure vi: MyDbApp: Test your db connection and login to your account

To connect MyDbApp with your database, it is however necessary to implement the backend using Java. For the frontend I've already created multiple templates. Using popup windows allows for a quick and easy insertion of new datasets.

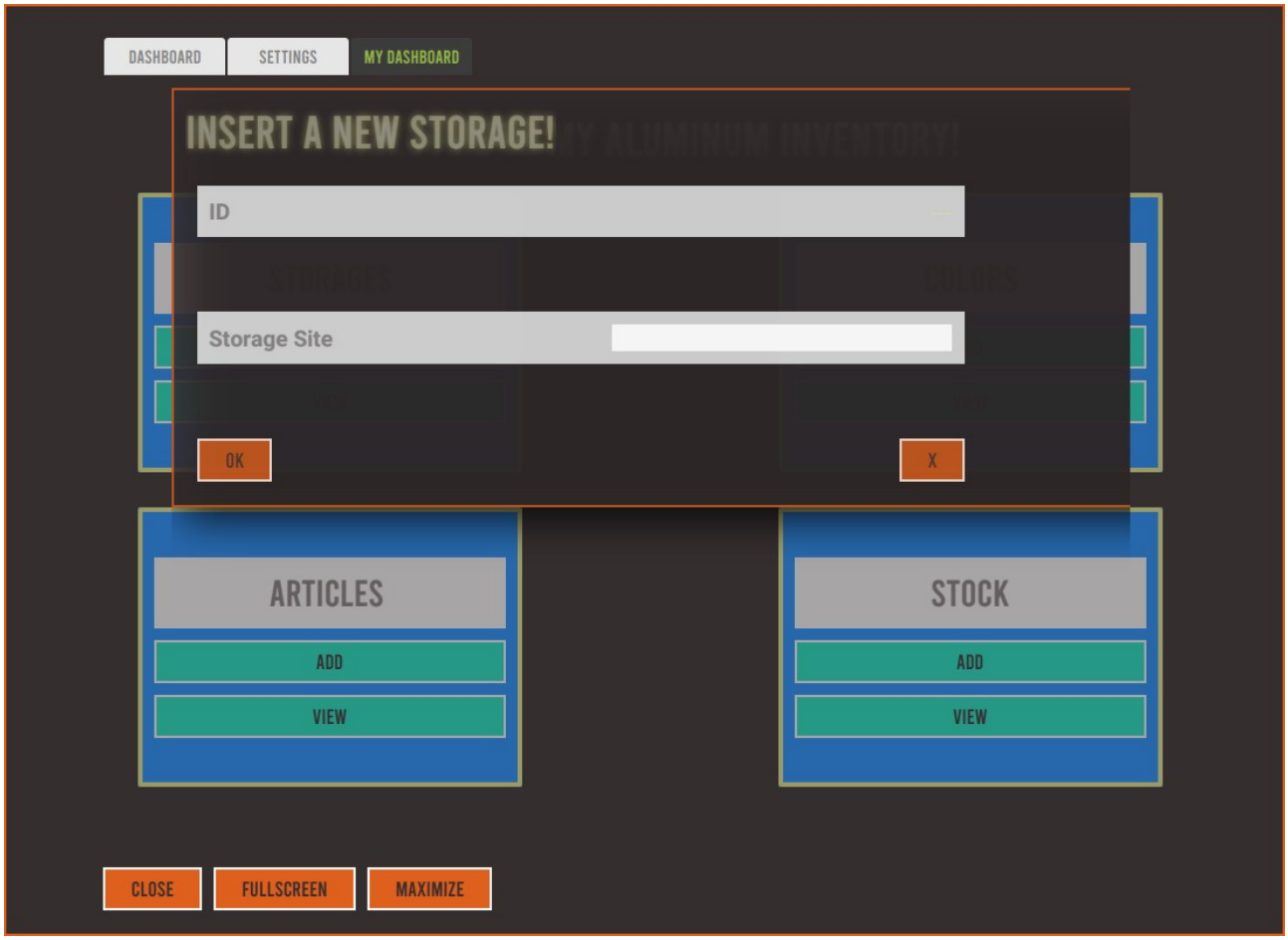

Figure vii: Popup window used for inserting new datasets

In the main view of one database entity you can usually delete, add or change datarows. What's more, I can implement additional buttons or functions for any databse entity.

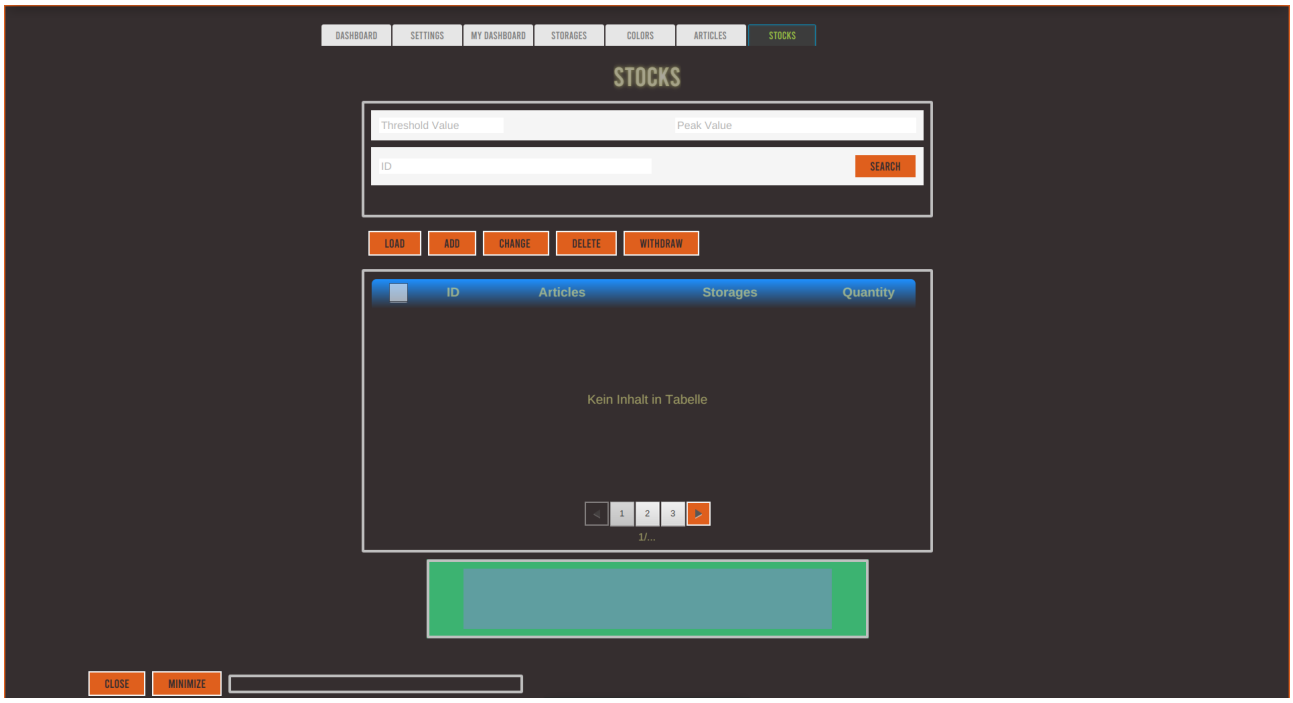

Figure viii: View designed to access the content of a database table

b.ii. Web Application

If you'd rather access your database via a web browser then I can implement the backend of your database app using **yesod**, a haskell framework. However, I can only connect an sqlite, mysql or postgresql database to the web application. The database queries will be mainly realized using the modules **Database.Persist** as well as **Esqueleto**. With Esqueleto I'm able to implement typesafe sql functions which extends the standard functions provided by Esqueleto.

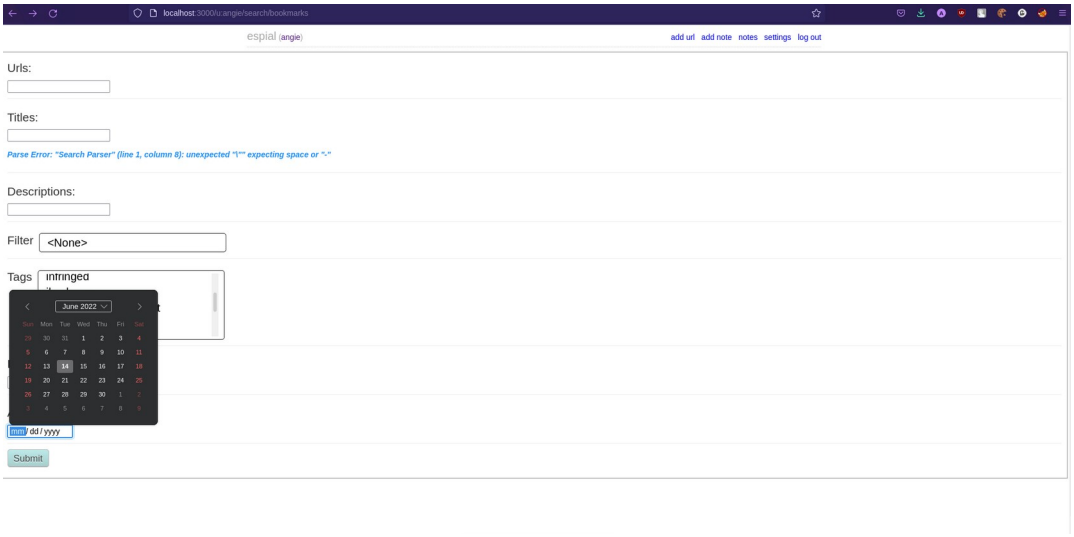

Figure ix: Search Form realized with Yesod

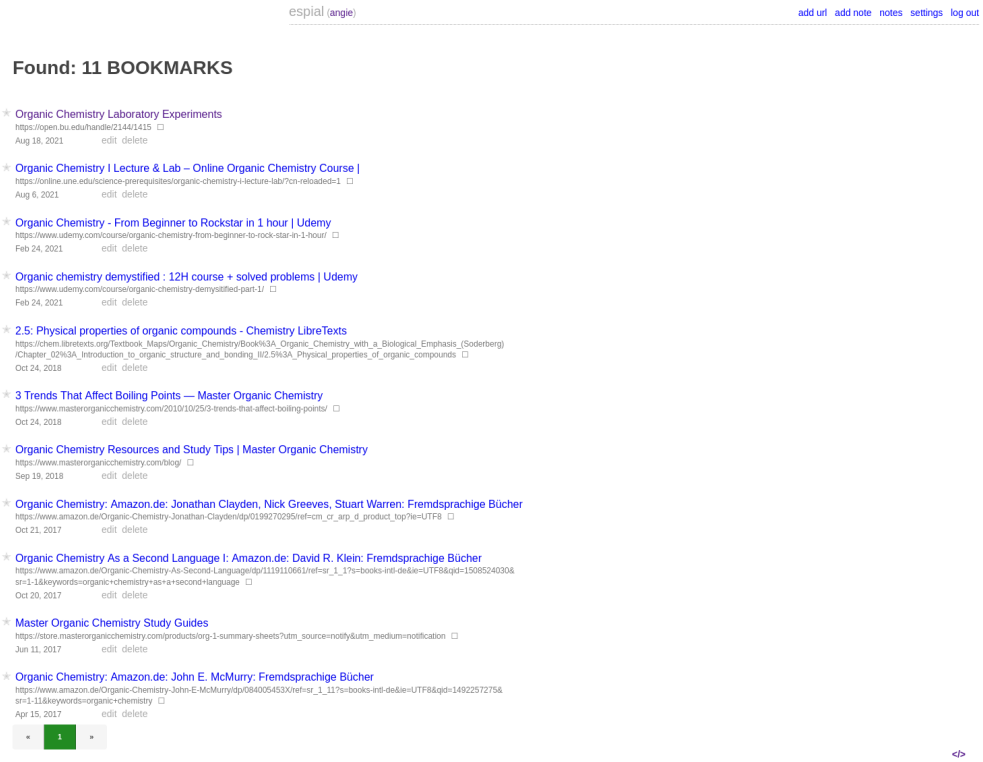

Figure x: View for jumping between the search result pages