Komponentenübersicht

Die Anwendung "Export Wizard" setzt sich aus drei Komponenten zusammen die alle lokal vorhanden sein müssen. Die auf MySQL basierende Datenbank des DidaktWizards, aus der die Daten für die PDF-Generierung stammen. Die Anwendung hat nur lesenden Zugriff auf diese Datenbank. Zweite Komponente ist eine sqlite-Datenbank, die für Datenhalten von Userdaten und Konfigurationstemplates zuständig ist. Die dritte Komponente ist die Desktop-Anwendung, über die Benutzer sich einloggen und PDFs zu einer Beruf/Ausbildungsjahr Kombination exportieren können. Die Desktopanwendung verbindet dabei die zwei Datenbanken mit der Geschäftslogik und stellt dafür ein GUI bereit.

Projektdaten

Das Projekt wurde in einer Cross-Platform Umgebung entwickelt und ist Platformunabhängig. Zum Einsatz kamen Windows, Linux und macOS Betriebssysteme.

MySQL Datenbank des Didakt-Wizard

MySQL ist ein Open-Source RDBMS die unter dem Schirm der Oracle-Corporation weiterentwickelt wird. Für die Datenabfragen wird die SQL-Syntax verwendet. Die MySQL-Datenbank wurde in einem Dockercontainer repliziert oder lokal installiert. Für die Anbindung über Dockercontainer wird die aktuelle Docker Version 17-CE benötigt. Für eine lokale Installation der mySQL-Datenbank wird die aktuelle Version der mySQL-Datenbank benötigt.

sqlite Datenbank

SQLite ist eine freie „Mini-SQL-Datenbank“, die im Gegensatz zu MySQL

oder PostgreSQL keinen laufenden Dienst benötigt. Klassische DBMS (Database Management Services) kommunizieren über Sockets oder IP mit Anwendungen und koordinieren deren Zugriffe auf die Datenbanken. Bei SQLite greifen Anwendungen direkt auf die Dateien mit den Datenbanken zu. Da es sich um eine Einzelplatzanwendung handelt ist sqlite als Datenbank ausreichend. sqlite wird in Version 3.19.3 benötigt.

Desktopanwendung

Übersicht

Die Desktopanwendung wurde in Java 8 entwickelt und wird als ausführbare .jar Datei ausgeliefert. Es handelt sich um eine JavaFX-Anwendung. Sie benötigt keine Installation.

Softwarearchitektur

Die Architektur wurde zuerst in zwei Teile aufgeteilt. Die Geschäftslogik bildet einen Teil und ist unabhängig von der GUI. Dies ermöglicht eine spätere Migration der Geschäftslogik zu einer Webanwendung. Die Geschäftslogik wurde in kleinere unabhängige Services unterteilt. Diese Kommunizieren über vorher definierte Interfaces.

Geschäftslogik

Die Geschäftslogik wurde in Repositories und ServiceProvider unterteilt und enthält die eigentliche Logik. Die ServiceProvider bieten Dienste für Datenbankconnections, PDF-Export, Authentifizierung und Authorisierung, Verwalten von Templates. Diese ServiceProvider bieten Interfaces und die konkreten Klassen implementieren die Funktionalität abhängig von den Anforderungen. Die ServiceProvider werden in den aufrufenden Klassen über Interfaces angesprochen. Die Repositories greifen über die ServiceProvider auf die Datenbanken zu und liefer Models oder Listen von Models zurück.

Entitäten

Report *Generelle Information über Report*

- String department
- Profession profession
- String teachingForm
- String director

Profession *Kombination von Beruf und Ausbildungsjahr*

- int id *Primary Key aus der Datenbank*
- String name *Berufname*
- int yearOfTraining *Ausbildungsjahr*
- List subjects *Liste der Lernbereiche*

Subject *Lernbereich*

- int id *Primary Key aus der Datenbank*
- String name *Berufname*
- List fields *Liste der Lernfelder*

Field *Lernfeld*

- int id *Primary Key aus der Datenbank*
- String name *Berufname*
- int duration *Dauer des Lernfeldes*
- List situations *Liste der Situations*

Situation *Lernsituation*

- int id *Primary Key aus der Datenbank*
- String name *Berufname*
- int duration *Dauer der Situation*
- int start *Anfangszeit*
- int end *Endzeit*
- String scenario *Lernszenario*

- String outcome *Handlungsprodukt*
- String competences *Lernkompetenzen*
- String content *Inhalt*
- String materials *Unterrichtsmaterialien*
- comments *Kommentar*
- List techniques *Liste der Lerntechniken*
- List achievments *Liste der Erfolge*

Technique *Lerntechnik*

- int id *Primary Key aus der Datenbank*
- String name *Berufname*

Achievment *Erworbene Kompetenz*

- int id *Primary Key aus der Datenbank*
- String name *Berufname*

Configuration *Konfigurationstemplate*

- int id *Primary Key aus der Datenbank*
- String name *Name des Templates*
- int userId *id des Users*
- bool scenario *Flag*
- bool outcome *Flag*
- bool competence *Flag*
- bool content *Flag*
- bool materials *Flag*
- bool comments *Flag*
- bool techniques *Flag*
- bool achievements *Flag*

Repositories

ReportSQLRepository *Für Datentransfer der ReportEntitäten*

- DBServiceProvider dbServiceProvider *Interface fuer DBService*
- ReportSQLRepository (DBServiceProvider dbsp) *CTOR mit DI*
- Report get(Profession p, int yearOfTraining)

ProfessionSQLRepository *Für Datentransfer der ProfessionEntitäten*

- DBServiceProvider dbServiceProvider *Interface fuer DBService*
- ProfessionSQLRepository (DBServiceProvider dbsp) *CTOR mit DI*
- List getAllProfessionsWithId()

SubjectSQLRepository *Für Datentransfer der SubjectEntitäten*

- DBServiceProvider dbServiceProvider *Interface fuer DBService*
- SubjectSQLRepository (DBServiceProvider dbsp) *CTOR mit DI*
- List getAllSubjectsForProfession(Profession p)

FieldSQLRepository *Für Datentransfer der FieldEntitäten*

- DBServiceProvider dbServiceProvider *Interface fuer DBService*
- FieldSQLRepository (DBServiceProvider dbsp) *CTOR mit DI*
- List getAllFieldsForSubject(Subject s)

SituationSQLRepository *Für Datentransfer der SituationEntitäten*

- DBServiceProvider dbServiceProvider *Interface fuer DBService*
- SituationSQLRepository (DBServiceProvider dbsp) *CTOR mit DI*
- List getAllSitutationsForField(Field f)
- List getAllAchievments(Situation s)
- List getAllTechniques(Situation s)

Services

DBServiceProvider T *Generisches Interface, das Connections zu einer Datenbank bereitstellt*

- T open() - T ist Connection

SQL2ODBServiceProvider *Konkrete Implementierung um über die SQL2O-Library Connections zu erstellen*

- Connection open()