**Fragen:**

**1.**Passt es wie ich es im TimeBooking Controller den erstellten TimeStamp dauerhaft gespeichert habe und ihn mit aktualisiere?

Passt es auch in der Person und bei UserContext, da habe ich die Person auch abgespeichert.

Damit man nicht dauernd die DB anfragen muss.

**2.**

Executor Service richtig benutzt bei Vacation Controller?

**3.**

Da ich ja mehrere Screens habe. Wäre da nicht gut ein Executor Service Provider?

Dann kann ich ihn in der JavaFX Shutdown Methode shutdownen.

Wenn ja muss ich goBack zentralisieren.

public class LoginView extends Application {  
 @Override  
 public void start(Stage stage) throws IOException {  
 try {  
 SceneManager.*setStage*(stage);  
 SceneManager.*switchScene*("/org/fabianandiel/gui/loginView.fxml", 400, 400, "Login");  
 } catch (IllegalArgumentException e) {  
 System.*out*.println(e.getMessage());  
 System.*exit*(0);  
 } catch (IOException e) {  
 System.*out*.println("IO Exception: " + e.getMessage());  
 System.*exit*(0);  
 }  
  
 }  
  
 @Override  
 public void stop() throws Exception {

HIER WÜRDE ICH DEN EXECUTOR SERVICE SHUTDOWNEN

EntityManagerProvider.*shutdown*();  
 }

UND in jedem Controller wo ich ihn brauche in der goBack to Main Methode().

**Erledigt: =====================================**

1. Ich würde gerne für den eingeloggten User eine public Singelton Klasse machen.

public class UserContext {

private static UserContext instance;

private String username;

private List<String> roles;

private UserContext() {

// Private constructor to prevent external instantiation

}

public static UserContext getInstance() {

if (instance == null) {

instance = new UserContext();

}

return instance;

}

public void initSession(String username, List<String> roles) {

this.username = username;

this.roles = roles;

}

public String getUsername() {

return username;

}

public List<String> getRoles() {

return roles;

}

public boolean hasRole(String role) {

return roles != null && roles.contains(role);

}

public void clearSession() {

this.username = null;

this.roles = null;

}

}

Und dann mit dem Stop hook bei den einzelnen Views, sollte der User auf die „brutale“ Abbrechen.

public class MainView extends Application {

@Override

public void start(Stage stage) {

// ...

}

@Override

public void stop() throws Exception {

// This will run when app exits

MyAppContext.cleanup();

}

}

PASST SO!

1. Was hältst du von dem private Konstruktor als extra Defense bei den Utility Klassen?

PASST!!!

1. Kann ich nicht so Singelton artig, wie mit dem eingeloggten User, auch im ersten auch den Entity Manager zur Verfügung stellen?

ZUR VERFÜGUNG STELLEN MIT SINGELTOM MUSTER.

1. Bei den Fragen was meinst du mit JavaFX Event Programmierung?

**27.06.2025:**

1. Wie bewertest du dass Klassendiagramm? Muss ich da unbedingt die Verbindungen rein machen?

Beim Klassendiagramm geht es um das ganz große Picture. Er will nicht die einzelnen Beans mit allen Feldern und Eigenschaften sondern, wie die großen Überkategorien miteinander reden. Wie schaut die ganze große Verbindung aus. Von GUI zu Controller zu DAO.

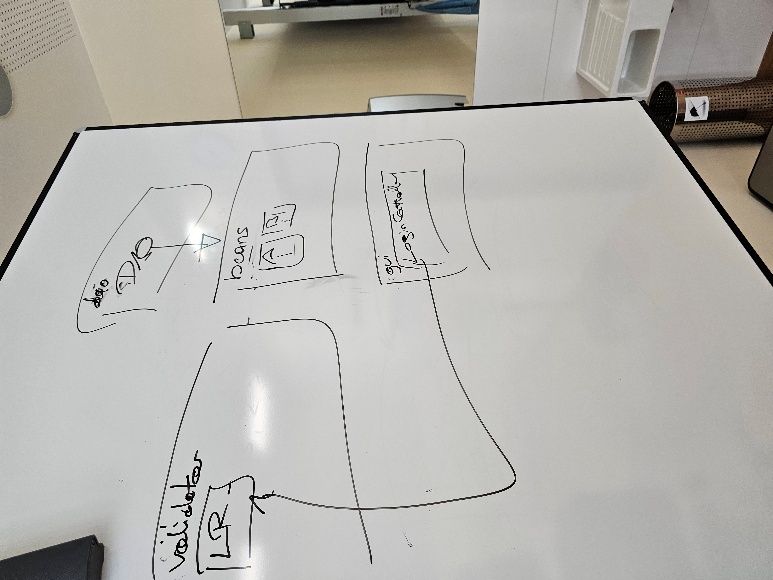
10 min erledigt.

Hier ist eher der Ablauf der Kommunikation. Grundkommunikation auf Package Ebene. Verbingung von was zu was. Nicht genau welche Methoden.

Hilfswerkzeug um Datenflow erklären zu können.

Ein Bild, das Text, Whiteboard, Zeichnung, Entwurf enthält.

Automatisch generierte Beschreibung



2. Situation LoginRequest:

Ist dann praktisch ein Validator Package. Ich transportiere nicht Sachen von A nach B. Package in Validator umbennnen.

Login Request gehört auch ins Klassendiagramm.

.

3. Wie findest du die Validierung mit Hibernate Validator?

Passt.

4. Wie findest du den Scene Manager? Gehört der mit dem Login Request ins Klassendiagramm?

Gehört auch ins Klassendiagramm spricht mit der GUI.

5. Ich bin darauf gekommen, dass sich folgenede Tabellen in meinem ERD eigentlich wie Enums verhalten:

Das kann ich aus der Datenbank rausnehmen.

6. Paßwörter mit Sternchen machen – password field

Vor der Kommission vertreten. Mit Sternchen austauschbar.

7. Warten noch mit dem File System. Am Anfang noch nicht mit Sprache zur Gänze ausnützen. Priorität erstmal ganz fertig werden.

8. Bei Hibernate kann ich auch die Datenbank schrittweise aufbauen. Legts mir automatisch immer wieder neu an.

Kleinigkeit wie Error Text setzen nicht in eigenen Thread. Größerer Bereich weil man Programm eher freezt.

**20.06.2025:**

* Kann ich auch die Manager und Admin Rolle zusammenlegen. So manche Manager haben diese andere nicht. Nur Manager dürfen die Admin Rolle haben.

Passt

* Ich würde gerne zip, city und country in eigenen Tabellen haben. Was denkst du?

Eine Beispieltabelle namens Addresse mit Stadt, Zip und Strasse,…. Mehrere an der selben Addresse

* Du willst ein Klassendiagramm und ERD richtig?

JA

* Passt es so wie ich es gemacht habe.

JA

* Leben die Dokumente bis ich das Projekt abgebe?

JA - jederzeit

* Ab wann siehst du es dir „richtig“ an?

Passt!

* Habe ich das so richtig gemacht zwischen pending request und person? Diese 2 Beziehungen?

Personen ID gehört als Antragsteller und Approver in den pending request und status.

Ein Bild, das Text, Diagramm, Reihe, Screenshot enthält.

Automatisch generierte Beschreibung

Und aus der Person:

Pending requests

Requests to approve

Linien auf Personen ID

Zeigen:

Draw.io ERD

Planung der DB in Word

Abgeben:

Commitment:

* Änderungen einarbeiten
* Klassendiagramm
* ERD fertig