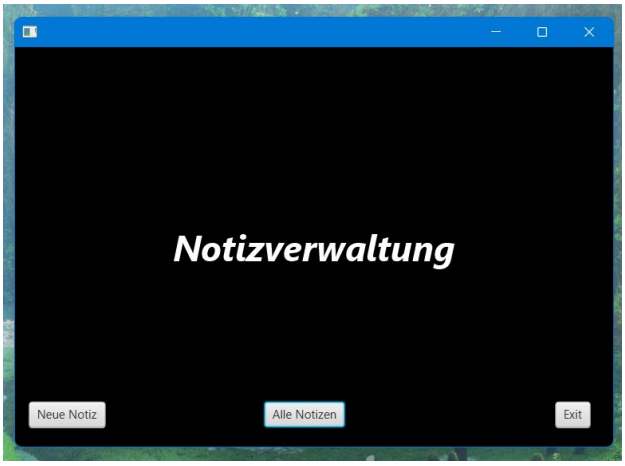
Notizverwaltung

Maximilian & Ajdin

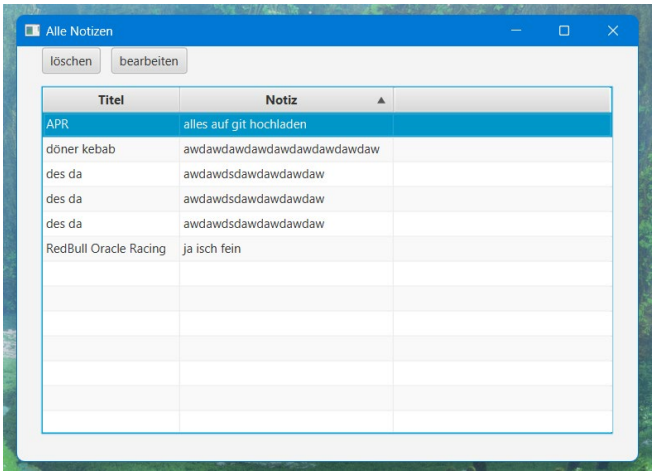
Unser Notizverwaltungsprogramm ermöglicht dem User eine Notiz zu erstellen und mit einem Titel und dem Content, welcher drinnen ist zu erstellen und der wird automatisch in der Datenbank gespeichert.
Wir haben mit einem Interface gearbeitet, und wir haben das Speichern in der Datenbank mithilfe von dem DAO-Pattern gemacht.
Das Ganze programmieren haben wir zusammen gemacht.



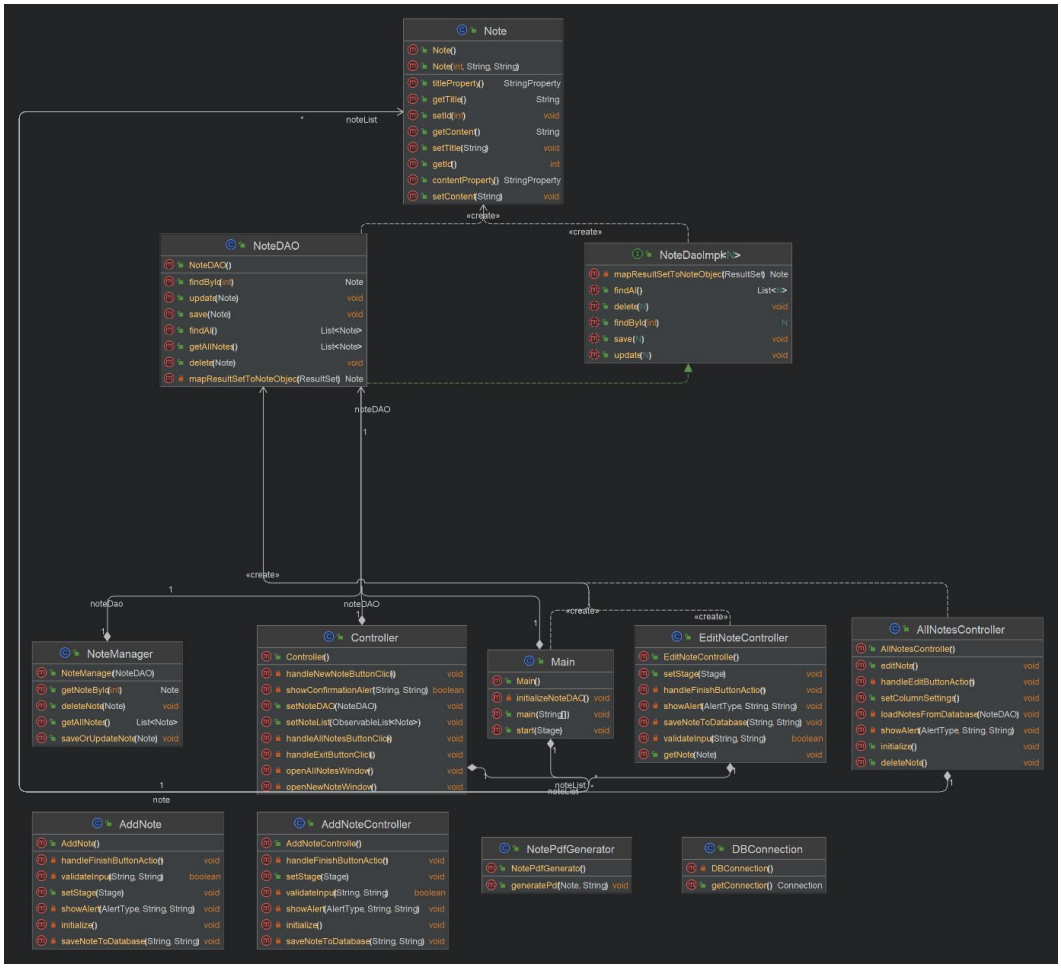
Wir begannen mit der Entwicklung unserer Klassen, Methoden und eines grundlegenden Layouts im JavaFX Scene Builder. Sobald wir eine grobe Vorstellung davon hatten, in welche Richtung das Projekt gehen sollte, haben wir auch die Datenbank dafür erstellt.

Nachdem wir alle Komponenten erfolgreich miteinander verknüpft hatten, haben wir unsere Methode zur Erstellung neuer Notizen getestet und festgestellt, dass alles einwandfrei funktioniert. Anschließend haben wir die "Alle Notizen.fxml"-Datei erstellt und den entsprechenden Controller sowie die Methoden implementiert. Sobald auch das einwandfrei funktionierte, wollten wir die Funktion zum Exportieren der Notizen als PDF umsetzen. Hierbei haben wir uns für den I-Text PDF Generator entschieden. Leider stießen wir dabei auf unerwartete Schwierigkeiten, wodurch wir gezwungen waren, beinahe das gesamte Projekt zu überarbeiten.

Auf der zweiten Seite finden sie das UML-Diagramm.



UML-Diagramm:



Analytische Rubrik:

	Codefunktionalität		Codequalität	
Punkte / Gewichtung	**Hier mit x ankreuzen**	**20**	**Hier mit x ankreuzen**	**30**
3 Punkte		Alle Funktionen lt. Spezifikation wurden implementiert und funktionieren fehlerfrei.	**x**	Der Code weist keine Codeduplikate auf bzw. setzt vollständig auf Wiederverwendung. Der Code ist durch neue Klassen erweiterbar, ohne dass das Restssystem geändert werden muss. Der Code ist damit sehr gut wartbar.
2 Punkte	**x**	Der Großteil der Funktionen lt. Spezifikation wurden implementiert und funktionieren weitgehend fehlerfrei.		Der Code weist kaum Codeduplikate auf bzw. setzt in weiten Teilen stark auf Wiederverwendung. Der Code ist durch neue Klassen erweiterbar, ohne dass größere Teile des Restssystems geändert werden müssen. Der Code ist weitgehend gut wartbar.
1 Punkt		Der Großtei der Funktionen lt. Spezifikation wurden nicht implementiert bzw. funktionieren nicht fehlerfrei.		Der Code weist einige Codeduplikate auf bzw. setzt das Prinzip der Wiederverwendung von Codeteilen kaum ein. Der Code ist durch neue Klassen nur erweiterbar, indem viele Teile des Restsystems angepasst werden müssen. Der Code ist schlecht wartbar.
0 Punkte		Keine der lt. Spezifikation geforderten Funktionen wurden korrekt implementiert.		Der Code weist viele Codeduplikate auf bzw. setzt das Prinzip der Wiederverwendung von Codeteilen nicht ein. Der Code ist durch neue Klassen nicht erweiterbar, ohne das Restsystem komplett umbauen zu müssen. Der Code ist nicht wartbar.
Summe	**2**		**3**	
Gewichtetes Ergebnis		**1,3**	**Prozent**	**87 %**