InvoiceManager

Silas Zahner | 29.06.2023

Inhaltsverzeichnis

Einleitung

Der Invoice Manager ist ein robustes Softwareprojekt, das entwickelt wurde, um die Verwaltung von Rechnungen und Produkten zu vereinfachen. Die Anwendung wurde in .NET C# unter Verwendung von WPF und EF Core implementiert und verwendet MSSQL als Backend-Datenbank.

Es gibt zwei Hauptgeschäftsobjekte in diesem System: "Invoice" und "Product". Die "Invoice" ist verantwortlich für die Verwaltung der Rechnungsdetails, einschliesslich des Datums und der damit verbundenen Produkte. Das "Product"-Objekt beinhaltet die Details der Produkte, die in der Rechnung aufgeführt sind, einschliesslich Namens, Beschreibung, Preis und Lagerbestand. Es gibt auch eine Datenanalyseklasse namens "AnalyticDto", die verschiedene Analyseinformationen bereitstellt, wie das meistverkaufte Produkt und die neueste Rechnung.

Zusätzlich zur grundlegenden CRUD-Funktionalität (Erstellen, Lesen, Aktualisieren, Löschen) für Rechnungen und Produkte bietet die Anwendung auch die Möglichkeit, verschiedene Arten von Datenanalysen durchzuführen. Dies ermöglicht es den Nutzern, wichtige Erkenntnisse und Trends aus ihren Rechnungs- und Produktinformationen zu gewinnen.

Wieso dieses Projekt?

Ich habe mich für dieses Thema entschieden, weil ich mich näher mit WPF (Windows Presentation Foundation) beschäftigen und meine ersten Schritte in diesem Bereich machen wollte. Bis jetzt habe ich meine Erfahrungen hauptsächlich mit WEB-APIs gesammelt und wollte daher meine erste native Windows-App erstellen.

Software-Entwicklungstool

Visual Studio und Azure DevOps wurden als primäre Entwicklungstools ausgewählt. Visual Studio ist eine leistungsfähige und umfangreiche IDE (Integrated Development Environment), die ideal für die Entwicklung von .NET-Anwendungen ist. Es bietet leistungsstarke Debugging-Werkzeuge, IntelliSense für die Code-Komplettierung und -Korrektur und unterstützt viele verschiedene Arten von Projekten und Dateiformaten. Darüber hinaus hat es eine enge Integration mit .NET und C#, was es zur idealen Wahl für dieses Projekt macht.

Azure DevOps wurde für das Projektmanagement und die Planung verwendet. Es bietet eine umfassende Suite von DevOps-Tools, die eine agile Projektplanung und -verfolgung, kontinuierliche Integration und Bereitstellung, Versionierungskontrolle und viele andere Features ermöglichen.

Die Benutzeroberfläche wird mit WPF (Windows Presentation Foundation) umgesetzt. Durch das Native Design kann so ein schnelles UI sichergestellt werden. Ausserdem bietet es eine nahtlose Integration mit EF Core und den anderen Tools die verwendet werden.

Als Codeverwaltungssystem wurde Git gewählt. Git ist ein weit verbreitetes und leistungsstarkes Tool für die Versionskontrolle. Es ermöglicht mehreren Entwicklern, gleichzeitig an einem Projekt zu arbeiten, ohne dass sie sich gegenseitig stören. Git speichert auch die Historie jeder Änderung, so dass es möglich ist, auf frühere Versionen des Codes zurückzugreifen, wenn nötig. Darüber hinaus hat es eine starke Integration mit sowohl Visual Studio als auch Azure DevOps, was die Entwicklungs- und Planungsprozesse nahtlos macht.

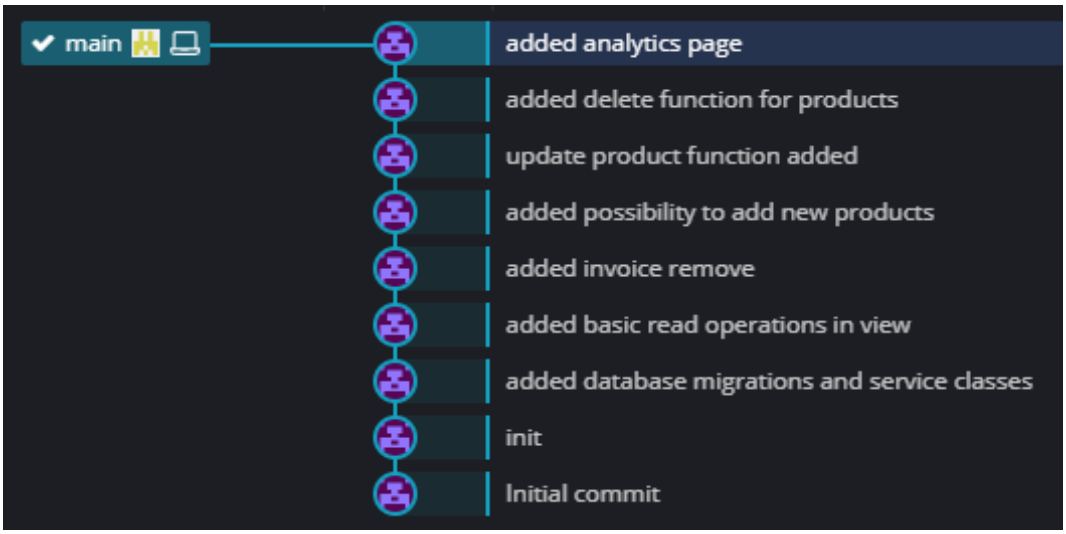

Abbildung 1: Versionsverlauf (GIT) in Gitkraken

Projektplanung

Projektmeilensteine und Termine

Die folgenden Meilensteine wurden für das Projekt festgelegt:

M1: Projektstart und Planung (3. Juni 2023)

Ziel dieses Meilensteins war es, das Projekt zu initiieren, die Anforderungen zu definieren und die Projektplanung durchzuführen.

M2: Design und Architektur (16. Juni 2023)

In dieser Phase wurden das Design und die Architektur der Software entwickelt. Darunter fällt auch die Definition der Datenbankstruktur und der Klassendiagramme.

M3: Implementierung (25. Juni 2023)

Dieser Meilenstein umfasste die eigentliche Codierung und Implementierung der Software gemäss den definierten Design- und Architekturvorgaben.

M4: Testing und Fehlerbehebung (28. Juni 2023)

Während dieses Meilensteins wurden umfangreiche Tests durchgeführt und identifizierte Fehler behoben.

M5: Projektabschluss (29. Juni 2023)

Schliesslich wurde die Software in die Produktionsumgebung übertragen und das Projekt offiziell abgeschlossen.

Die Termine wurden so festgelegt, dass ausreichend Zeit für alle Phasen des Projekts zur Verfügung steht und das Projekt pünktlich zum Abgabetermin am 29. Juni 2023 abgeschlossen wird.

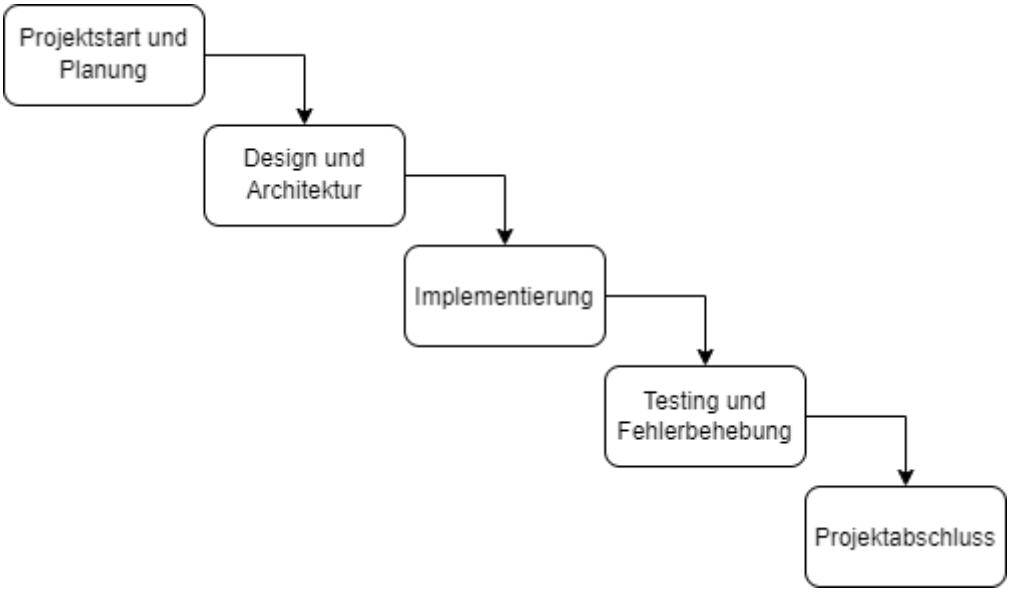

Abbildung 2: Projektplanung, aufzeigung der Meilensteine

Architektur und Design

Logische Architektur und Schichtentrennung

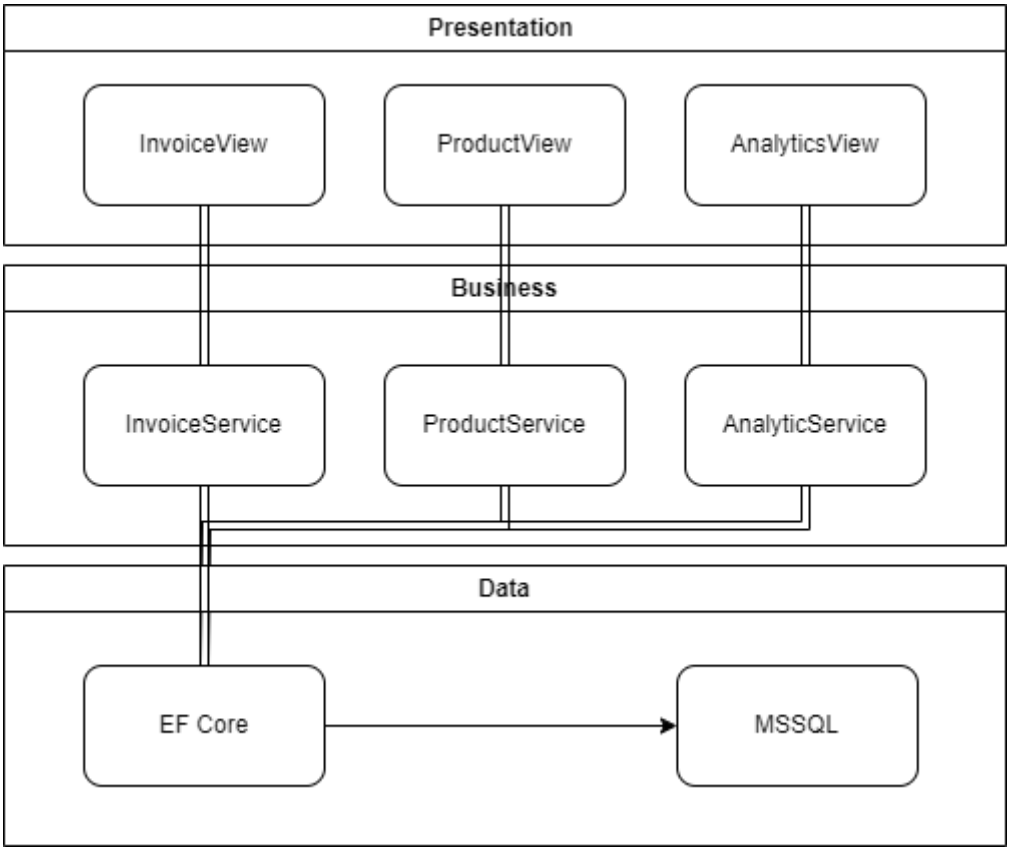

Abbildung 3: Schichtenarchitektur

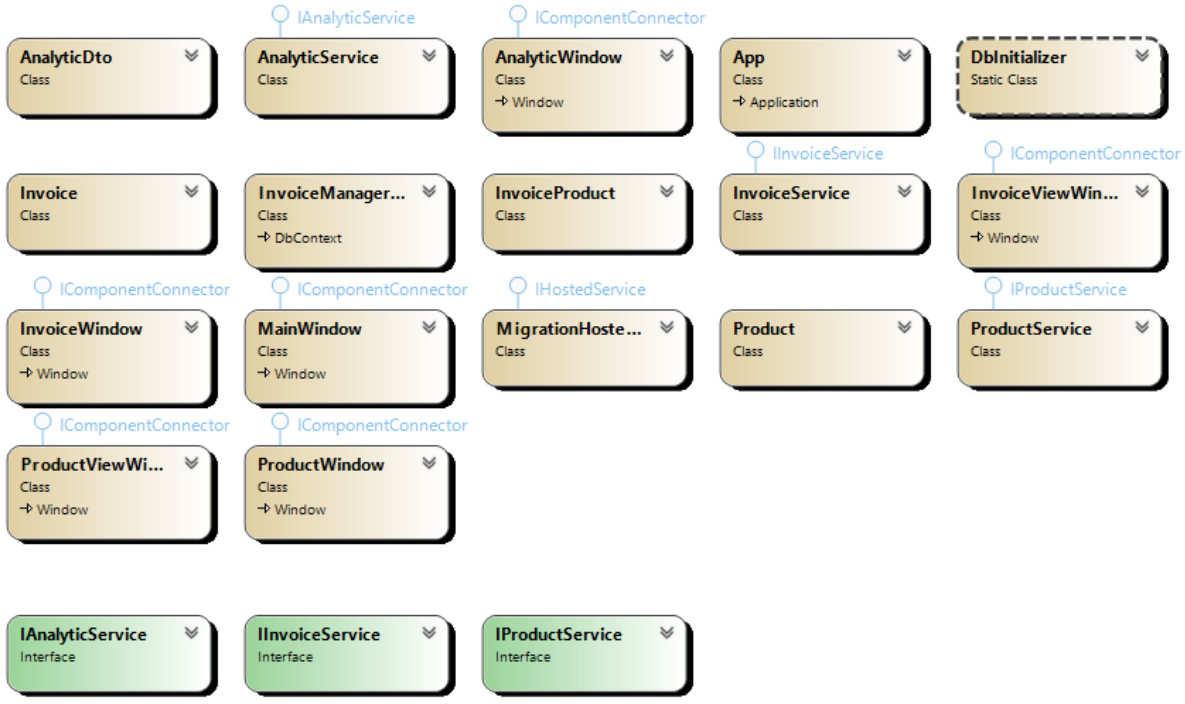

Abbildung 4: Klassendiagramm

Softwarekomponenten und ihre Verantwortlichkeiten

Die Hauptsoftwarekomponenten und ihre Verantwortlichkeiten sind:

Invoice: Verwaltet die Rechnungsdaten und die Beziehung zu den zugehörigen Produkten.

Product: Verantwortlich für die Verwaltung der Produktinformationen.

Analytic: Bietet verschiedene Analyseinformationen, wie das meistverkaufte Produkt und die neueste Rechnung.

Diese 3 Objekte sind der Kern der Applikation. Die Service Klassen Interagieren Dabei mit Ihnen.

Schnittstellen

Ich habe mich für diese Variante entschieden, weil sie eine klare Trennung zwischen Frontend und Backend bietet. Durch die Nutzung von WPF können wir die nativen Funktionen von Windows nutzen, was uns eine flexible Desktop-Anwendung und eine wiederverwendbare Codebasis verschafft. Allerdings beschränkt uns die Verwendung von WPF auf Windows-Geräte. Das Design stellt für mich persönlich eine zusätzliche Herausforderung dar, da das XML-Styling sich von den meisten UI-Bibliotheken unterscheidet. Insgesamt bietet sie jedoch eine robuste und skalierbare Lösung für das Projekt.

Datenspeicherung und Datenbankdiagramm

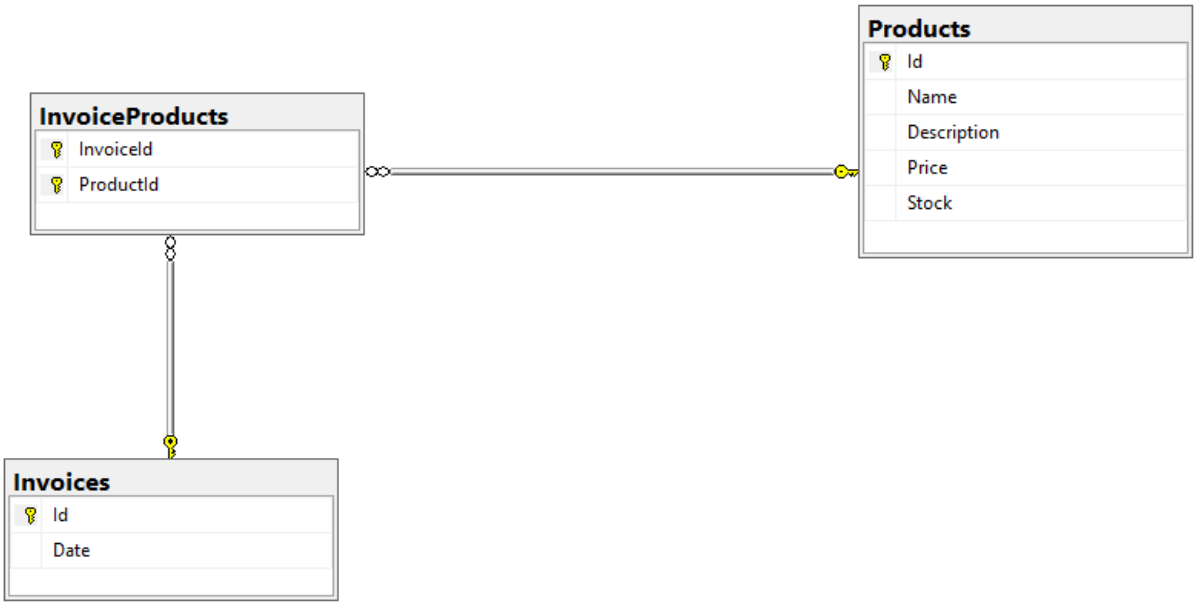

Abbildung 5: ER-Diagramm

Die Anwendung verwendet MSSQL als Backend-Datenbank, die zusammen mit EF Core die Speicherung und Abrufung von Daten aus der Datenbank ermöglicht. Die Datenbank enthält Tabellen für Rechnungen und Produkte, und die Beziehungen zwischen diesen werden durch EF Core verwaltet.

Anforderungsspezifikationen

Systemkontext

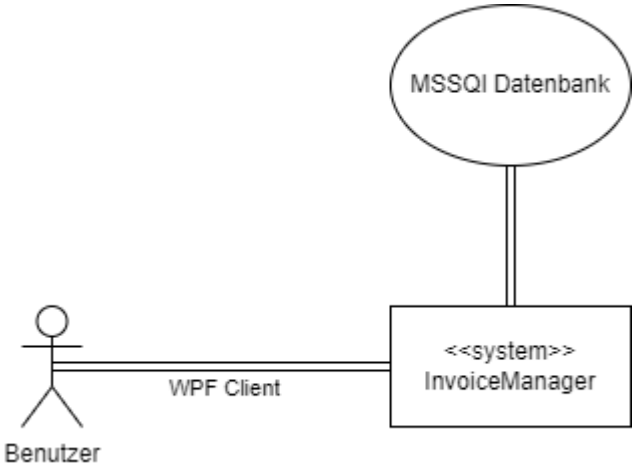

Abbildung 6: Systemkontext

Funktionale Anforderungen

Die funktionalen Anforderungen des Invoice Managers umfassen:

- **F1 - Rechnungserstellung:** Der Benutzer sollte in der Lage sein in unter 1min, eine neue Rechnung zu erstellen und diese mit den entsprechenden Produkten zu verknüpfen.
- **F2 - Produktverwaltung:** Der Benutzer sollte in der Lage sein in unter 1min, Produkte zu erstellen, zu lesen, zu aktualisieren und zu löschen (CRUD-Operationen).
- **F3 - Datenanalyse:** Die Anwendung sollte verschiedene Analysen bereitstellen, wie das meistverkaufte Produkt, die neueste Rechnung und das Produkt, das aus dem Bestand genommen wurde. Der Benutzer soll die Daten in unter 1min sehen.

Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen des Invoice Managers umfassen:

- **NF 1 - Benutzerfreundlichkeit:** 80% der Benutzer sollten die Anwendung ohne Schulung bedienen können.
- **NF 2 - Leistung:** Die Anwendung sollte auf Benutzeraktionen in weniger als 2 Sekunden reagieren.
- **NF 3 - Zuverlässigkeit:** Die Anwendung sollte eine Betriebszeit von 99,9% haben.

Use-Case-Diagramm

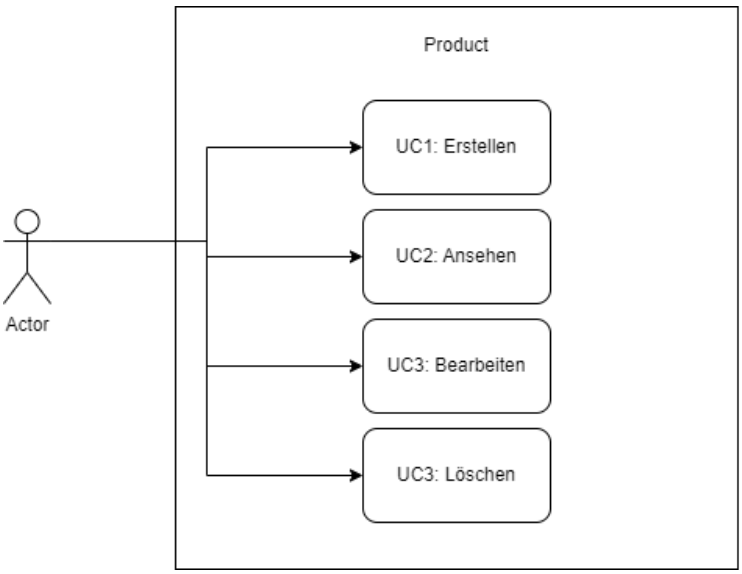

Abbildung 7: Use-Case-Diagramm Produkt

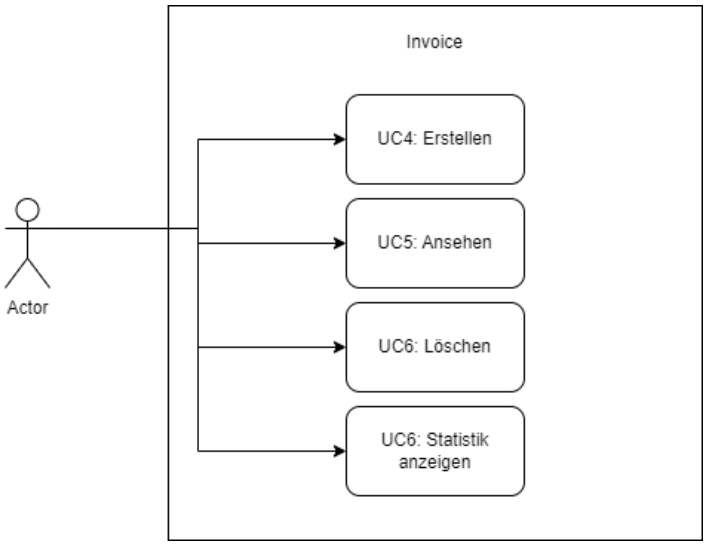

Abbildung 8: Use-Case-Diagramm Rechnung

Tabelle 1: Beschreibung Use-Case 1

Beschreibung	**UC1: Der Benutzer kann ein neues Produkt erstellen mit Namen, Beschreibung, Preis und Lageranzahl**
Akteure	Benutzer
Auslöser	Der Benutzer über das UI (Product => new)
Vorbedingung	• Der Benutzer hat die Applikation gestartet. • Es sind alle relevante Daten durch den Benutzer eingegeben
Normalablauf	• Der Benutzer öffnet die App • Klickt im Menu auf «Product» dann «new» • Gibt die erforderlichen Daten ein • Klickt auf «Submit»

Tabelle 2: : Beschreibung Use-Case 2

Beschreibung	**UC4: Der Benutzer kann eine neue Rechnung erstellen mit beliebig vielen Produkten**
Akteure	Benutzer
Auslöser	Der Benutzer über das UI (Invoice => new)
Vorbedingung	• Der Benutzer hat die Applikation gestartet. • Es sind alle relevante Daten durch den Benutzer eingegeben
Normalablauf	• Der Benutzer öffnet die App • Klickt im Menu auf «Invoice» dann «new» • Wählt eins, oder mehrere Produkte aus, indem er sie markiert. • Klickt auf «Submit»

Sequenzdiagramm

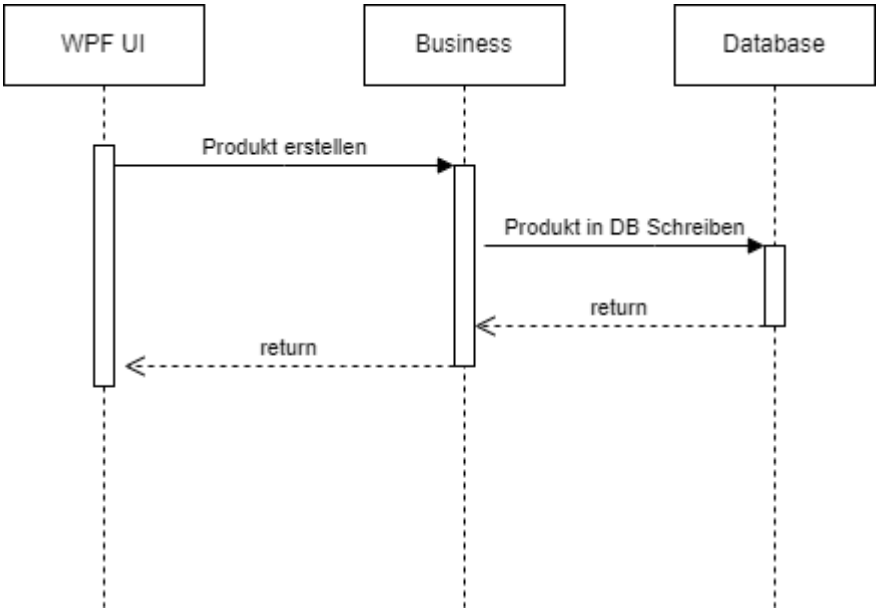

Abbildung 9: Sequenzdiagramm Produkt erstellen

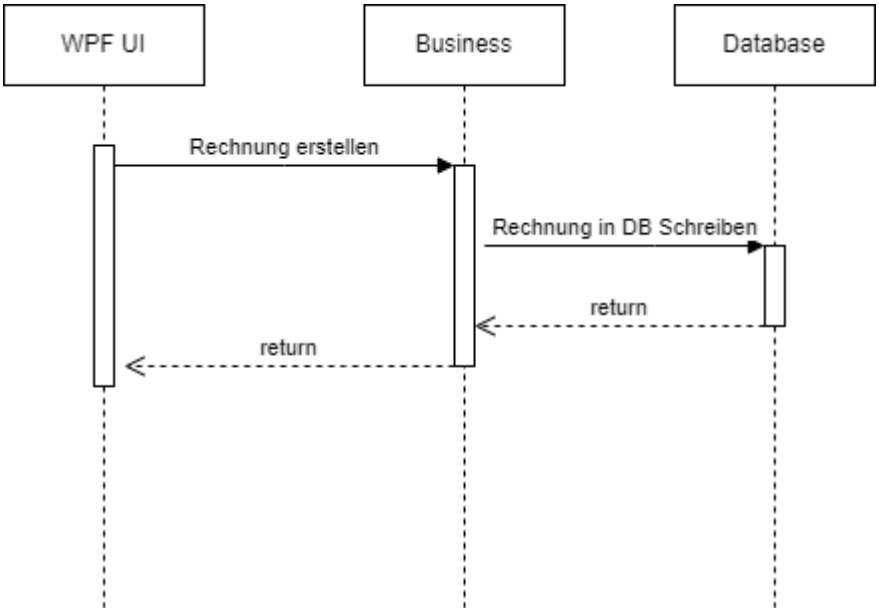

Abbildung 10: Sequendiagramm Rechnung erstellen

UI-Design und Interaktionen

Interaktionen zwischen den Steuerelementen

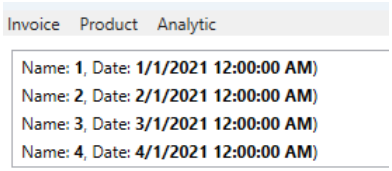

Abbildung 11: Startscreen, alle Rechnungen

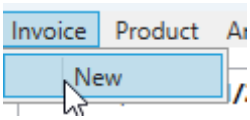

Abbildung 12: Neue Rechnung erstellen

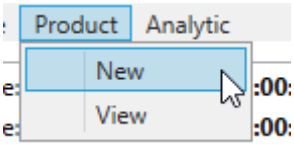

Abbildung 13: Neues Produkt erstellen bzw. alle anzeigen

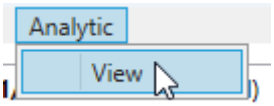

Abbildung 14: Analytics anzeigen

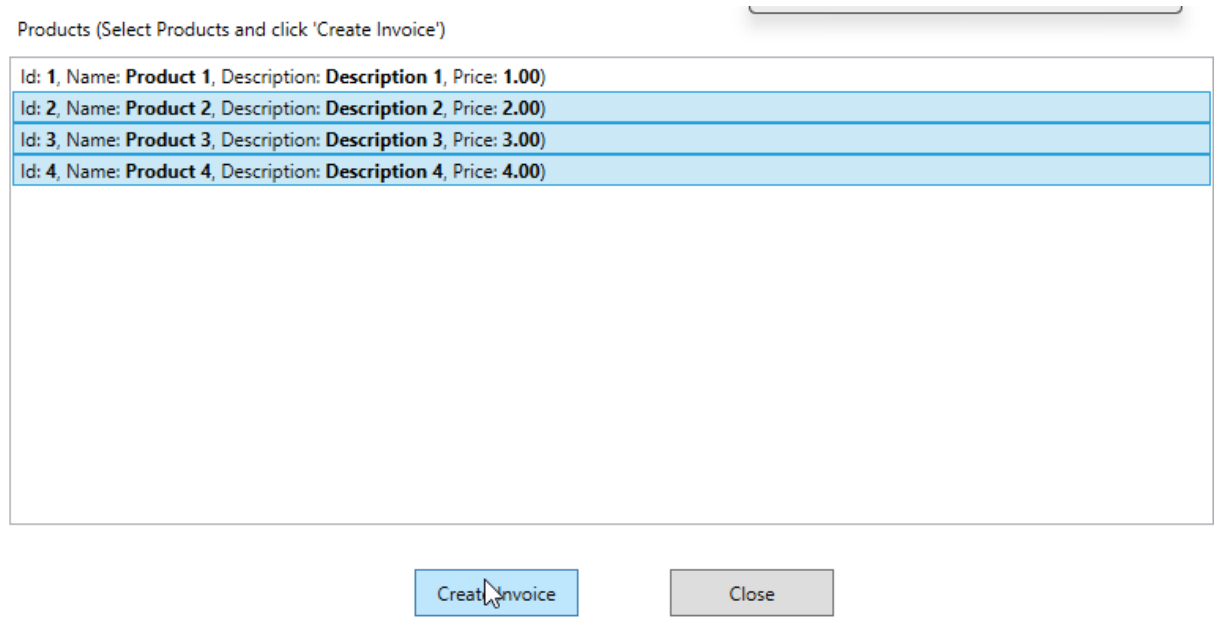

Abbildung 15: Produkte markieren um Rechnung zu erstellen

Abbildung 16: Produkt erstellen/anpassen

Id: **1**, Name: **Product 1**, Description: **Description 1**, Price: **1.00**)
Id: **2**, Name: **Product 2**, Description: **Description 2**, Price: **2.00**)
Id: **3**, Name: **Product 3**, Description: **Description 3**, Price: **3.00**)
Id: **4**, Name: **Product 4**, Description: **Description 4**, Price: **4.00**)

Abbildung 17: Alle Produkte

Most Sold Product	Product 1
Newest Invoice	01.04.2021
Out Of Stock Product	Product 4,
Most Products In One Invoice	2

Abbildung 18: Analytics

Tests und Testprotokolle

Funktionale Tests

Tabelle 3: Test (TF01)

Testfallnummer	**TF01**
Beschreibung	Erstellen einer neuen Rechnung
Ablauf	1. Öffnen der Anwendung 2. Klicken im Menü auf «Invoice» dann «new» 3. Ausfüllen der erforderlichen Felder mit gültigen Daten und Speichern der Rechnung
Erwartetes Ergebnis	Die Rechnung wird erfolgreich in der Datenbank gespeichert und in der Rechnungsliste angezeigt.
Erfolgskriterium	Alle Schritte werden erfolgreich durchgeführt und die Rechnung wird erfolgreich erstellt.

Tabelle 4: Test (TF02)

Testfallnummer	**TF02**
Beschreibung	Aktualisierung eines Produkts
Ablauf	1. Öffnen der Anwendung 2. Klicken im Menü auf «Invoice» dann «view» 3. Doppelklick auf das zu bearbeitende Produkt 4. Aktualisieren der Produktinformationen mit neuen gültigen Daten und Speichern der Änderungen
Erwartetes Ergebnis	Die Produktinformationen werden erfolgreich in der Datenbank aktualisiert und die Änderungen werden angezeigt.
Erfolgskriterium	Alle Schritte werden erfolgreich durchgeführt und das Produkt wird erfolgreich aktualisiert.

Tabelle 5: Test (TF03)

Testfallnummer	TF03
Beschreibung	Durchführen einer Datenanalyse
Ablauf	1. Öffnen der Anwendung 2. Klicken im Menü auf «Analytic» dann «view»
Erwartetes Ergebnis	Die Anwendung führt die Datenanalyse durch und zeigt die entsprechenden Ergebnisse an.
Erfolgskriterium	Alle Schritte werden erfolgreich durchgeführt und die Datenanalyseergebnisse werden erfolgreich angezeigt.

Tabelle 6: Test (TF04)

Testfallnummer	TF04
Beschreibung	Löschen einer Rechnung
Ablauf	1. Öffnen der Anwendung 2. Rechtsklick auf die gewünschte Rechnung in der Liste 3. Klicken auf «Delete»
Erwartetes Ergebnis	Die ausgewählte Rechnung wird erfolgreich aus der Datenbank gelöscht und aus der Rechnungsliste entfernt.
Erfolgskriterium	Alle Schritte werden erfolgreich durchgeführt und die Rechnung wird erfolgreich gelöscht.

Testprotokoll

Tabelle 7: Testprotokoll, die ausführung der Tests

Testfallnummer	Zeitpunkt	Tester	Erwartetes Ergebnis	Tatsächliches Ergebnis	Abweichungen
TF01	28.06.2023	Silas	Die Rechnung wird erfolgreich erstellt und in der Datenbank gespeichert	Die Rechnung wurde erfolgreich erstellt und in der Datenbank gespeichert	Keine
TF02	28.06.2023	Silas	Das Produkt wird erfolgreich aktualisiert	Das Produkt wurde erfolgreich aktualisiert	Keine
TF03	28.06.2023	Silas	Die Datenanalyseergebnisse werden erfolgreich angezeigt	Die Datenanalyseergebnisse wurden erfolgreich angezeigt	Keine
TF04	28.06.2023	Silas	Die Rechnung wird erfolgreich gelöscht	Die Rechnung wurde erfolgreich gelöscht	Keine

In den Testprotokollen werden die Ergebnisse der durchgeführten Tests detailliert festgehalten. Für jeden Testfall wurde das erwartete Ergebnis mit dem tatsächlichen Ergebnis verglichen, und alle Abweichungen wurden festgehalten und analysiert. Alle in den Tests identifizierten Fehler wurden behoben.

Schlussfolgerungen und zukünftige Arbeit

Schlussfolgerungen:

Das Invoice Manager Projekt wurde erfolgreich umgesetzt und erfüllt die funktionalen Anforderungen zur Verwaltung von Rechnungen und Produkten. Das UI-Design ist benutzerfreundlich und ermöglicht eine intuitive Interaktion. Die Implementierung basiert auf einer soliden Architektur mit klar definierten Softwarekomponenten und einer effizienten Datenspeicherung. Die funktionalen Tests wurden durchgeführt und Fehler wurden behoben, um eine stabile Funktionalität sicherzustellen.

Zukünftige Arbeit:

Der Invoice Manager bietet eine gute Grundlage für zukünftige Erweiterungen und Verbesserungen. Zum Beispiel könnte das Inventarmanagement durch die Verwaltung des Lagerbestands erweitert werden. Weitere Funktionalitäten wie erweiterte Suchfunktionen, Benutzerauthentifizierung und Autorisierung, Berichterstattung und Optimierungen der Leistung könnten implementiert werden. Die Anwendung kann auch auf andere Plattformen oder Gerätetypen erweitert werden, um eine breitere Nutzerbasis zu erreichen.

Ausblick:

Der Invoice Manager bietet eine solide Lösung für die Verwaltung von Rechnungen und Produkten. Mit der Möglichkeit zur Erweiterung und Verbesserung bietet er ein grosses Potenzial, den Anforderungen und Bedürfnissen der Benutzer gerecht zu werden. Durch kontinuierliche Aktualisierungen und Feedback der Benutzer kann der Invoice Manager zu einer noch leistungsfähigeren und umfassenderen Lösung entwickelt werden, die eine effiziente Verwaltung von Rechnungen und Produkten ermöglicht.

Anhänge

Glossar

Tabelle 8: Glossar

Begriff	Beschreibung
Invoice Manager	Der Invoice Manager ist die entwickelte Softwareanwendung zur Verwaltung von Rechnungen und Produkten.
CRUD	Eine Abkürzung für Create, Read, Update, Delete. Es bezieht sich auf die grundlegenden Operationen zum Erstellen, Lesen, Aktualisieren und Löschen von Daten.
UI	Eine Abkürzung für User Interface (Benutzeroberfläche). Es bezieht sich auf den visuellen Teil einer Anwendung, mit dem Benutzer interagieren können.
EF Core	Eine Abkürzung für Entity Framework Core. Es handelt sich um ein Open-Source-ORM-Framework (Object-Relational Mapping) von Microsoft, das die Datenbankkommunikation und den Objektzugriff in .NET-Anwendungen erleichtert.
WPF	Eine Abkürzung für Windows Presentation Foundation. Es handelt sich um ein Framework zur Entwicklung von Windows-Desktopanwendungen mit einer flexiblen Benutzeroberfläche.
Azure DevOps	Eine Sammlung von Entwicklungstools und -diensten, die eine vollständige End-to-End-Softwareentwicklung und -bereitstellung ermöglichen.
ER-Diagramm	Ein Entity-Relationship-Diagramm, das die Beziehungen zwischen Entitäten in einer Datenbank darstellt.
MSSQL	Eine Abkürzung für Microsoft SQL Server. Es handelt sich um ein relationales Datenbankverwaltungssystem von Microsoft.

Abbildungsverzeichnis

Tabellenverzeichnis