

Zentrum für berufliche Weiterbildung, ZbW

Projektarbeit – Auftragsverwaltung

Dozent:
Thomas Kehl
&
Larissa Fitz

Autoren:
Elias Götte
Khabat Rammo
Maximilian Degen

Klasse IT-A 22H-25H

Inhaltsverzeichnis

Literaturverzeichnis	3
Tabellenverzeichnis	4
Abbildverzeichnis	4
Arbeitsjournal.....	5
Organisation.....	7
Projektmitglieder	7
Allgemein.....	7
Zeitplan	8
Aufgabenaufteilung	9
Einführung.....	10
Methodik.....	11
Scrum.....	11
Sprints.....	11
Wöchentliche Stand-up-Meetings.....	11
Sprint-Planung	11
Sprint-Review.....	11
Retrospektiven	11
DevOps	11
Board zur Aufgabenverwaltung	11
Repos für Versionskontrolle	11
Pipelines für Build und Release.....	11
Testpläne	11
GitHub	12
Versionskontrolle mit GitHub	12
Code-Reviews und Zusammenarbeit.....	12
Integration mit Azure DevOps.....	12
Fazit	12
Agile Durchführung und Sprint Management	12
Sprint Backlog und Task Breakdown	12
Sprint Review und Retrospektive.....	13
Sprint 0: Projekteinrichtung	13
Sprint 1: Ausführung und Überprüfung	14
Sprint 2: Details zu Aufgaben und Fortschritt.....	14
Sprint 3: Aufgabenstatus und Handlungsbedarf	15
Sprint 4: Überprüfung und Fortschritt.....	15
Sprint 5: Planung und Erwartungen.....	15
Initialisierung Azure DevOps [3]	16
Schätzung des Produkt-Backlogs.....	16

Durchführung der Schätzung:	16
User Stories Sammlung:.....	16
Einführung in die Methode.....	16
Diskussion der User Stories	16
Abstimmung über WhatsApp.....	16
Auswertung der Ergebnisse	17
Entscheidungsfindung	17
Rationale für Schätzwerte	17
Git-Management und Sourcecode-Verwaltung	17
Branching und Merging Strategie	17
Commit Konvention und Tags	18
Integration in die Projektumgebung	18
Continuous Integration und Automatisierte Tests	18
Aufbau der Release-Pipeline	19
Feedback-Mechanismen und Fehlerbehandlung	19
Bedeutung der CI für das Projekt.....	19
Datenbank- und Anwendungsfunktionen.....	20
Datenbankmodell	20
Analyse der Commit-Historie	20
UI/UX	21
Inbetriebnahme der Applikation und Generierung der Datenbank.....	22
Voraussetzungen:	22
Schritte zur Einrichtung der Datenbank:	22
Generierung von Testdaten:.....	22
Daten Seeder:	22
Beispieldaten:.....	22
Starten der Applikation:	22
Wichtige Code-Komponenten:.....	22

Literaturverzeichnis

- [1] it-agile, "scrum," [Online]. Available: <https://www.it-agile.de/agiles-wissen/scrum/was-ist-scrum/>. [Accessed 2024].
- [2] microsoft, "what-is-devops," [Online]. Available: <https://azure.microsoft.com/de-de/resources/cloud-computing-dictionary/what-is-devops>. [Accessed 2024].
- [3] microsoft, "devops," [Online]. Available: <https://learn.microsoft.com/de-de/azure/devops/organizations/projects/create-project?view=azure-devops&tabs=browser>. [Accessed 2024].
- [4] J. Martins, "asana.com," [Online]. Available: <https://asana.com/de/resources/t-shirt-sizing>. [Accessed 2024].
- [5] M. REHKOPF, "atlassian," [Online]. Available: <https://www.atlassian.com/de/agile/project-management/user-stories>. [Accessed 2024].
- [6] atlassian, "merge-strategy," [Online]. Available: <https://www.atlassian.com/de/git/tutorials/using-branches/merge-strategy>. [Accessed 2024].
- [7] atlassian, "atlassian," [Online]. Available: <https://www.atlassian.com/git/tutorials/inspecting-a-repository/git-tag#:~:text=Tags%20are%20ref's%20that%20point,branch%20that%20doesn't%20change..> [Accessed 2024].
- [8] atlassian, "continuous-integration," [Online]. Available: <https://www.atlassian.com/de/continuous-delivery/continuous-integration>. [Accessed 2024].
- [9] S. Krug, "Don't Make Me Think".
- [10] microsoft. [Online]. Available: <https://learn.microsoft.com/de-de/ef/core/get-started/overview/install>. [Accessed 2024].
- [11] microsoft, "sql-server-downloads," [Online]. Available: <https://www.microsoft.com/en-us/sql-server/sql-server-downloads>. [Accessed 2024].
- [12] openai. [Online]. Available: <https://chat.openai.com/>. [Accessed 2024].

Tabellenverzeichnis

Tabelle Arbeitsjournal	7
Tabelle Projektmitglieder.....	7
Tabelle Allgemein	8
Tabelle Zeitplan	9
Tabelle Sprint 0.....	14
Tabelle Sprint 01.....	14
Tabelle Sprint 02.....	15
Tabelle Sprint 03.....	15
Tabelle Sprint 04.....	15
Tabelle Sprint 05.....	16

Abbildverzeichnis

Abbildung 1 Test run	12
Abbildung 2 Sprint 0.....	13
Abbildung 3 Schätzung des Produkt-Backlogs.....	17
Abbildung 4 Schätzung des Produkt-Backlogs.....	17
Abbildung 5 Branching und Merging Strategie	18
Abbildung 6 Pipeline	19
Abbildung 7 Feedback-Mechanismen	19
Abbildung 8 GUI.....	21

Arbeitsjournal

Version	Datum	Effektiver Aufwand [h]	Aufwand in Gruppe [h]	Anz. Mitgl.	Tätigkeit	Mitglieder
1.1	08.01.2024	6h	18h	3	Azure DevOps kennenlernen	EG, KR, MD
1.2	08.01.2024	1h	3h	3	Initial Project Setup in Azure DevOps	EG, KR, MD
1.3	08.01.2024	2h	6h	3	Sprintplanung und Aufgabenzerlegung	EG, KR, MD
1.4	09.01.2024	1h30	4h30	3	User Story Definition	EG, KR, MD
1.5	10.01.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.6	13.01.2024	1h	-	1	GitHub Repository erstellen	EG
1.7	14.01.2024	1h30	-	1	Entwurf der Dokumentationsstruktur	KR
1.8	15.01.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.9	17.01.2024	2h	-	1	Planung des Dokumentationsinhalts	KR
1.10	18.01.2024	1h	3h	3	Erstellung von Tags auf GitHub	EG, KR, MD
1.11	20.01.2024	2h	-	1	Pipeline & GitHub einrichten	EG
1.12	21.01.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.13	23.01.2024	2h	6h	3	Diskussion über Architektur und DB-Design	EG, KR, MD
1.14	24.01.2024	1h	-	1	Planung der DB und zukünftiger Arbeiten	KR
1.15	27.01.2024	1h	-	1	Modelle entwickeln	KR
1.16	28.01.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.17	30.01.2024	1h	2h	2	Datenbank erstellen	MD, EG
1.18	31.01.2024	3h30	7h	2	Repository entwickeln, Datenbank aktualisieren	EG, MD
1.19	03.02.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.20	04.02.2024	1h	-	1	Entwicklung des Rechnungsmodells und DB	EG
1.21	04.02.2024	2h	4h	2	CTE Query für hierarchische Artikelgruppenverwaltung	MD, EG
1.22	07.02.2024	2h30	5h	2	Daten Seeding entwickeln	MD, EG
1.23	09.02.2024	6h	18h	3	Entwicklung von CTE	MD, EG, KR

1.24	09.02.2024	2h	4h	2	Dokumentation für DBA-Teilaufgaben	MD, EG
1.25	11.02.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.26	16.02.2024	4h	-	1	Versuch, temporale Tabellen hinzuzufügen	MD
1.27	17.02.2024	6h	-	1	Weiterarbeit an temporalen Tabellen	EG
1.28	18.02.2024	2h	-	1	Temporale Tabellen abschliessen	KR
1.29	19.02.2024	2h	-	1	Entwicklung des temporalen Repositories	EG
1.30	19.02.2024	4h	8	2	Design der Rechnungs-UI	EG, KR
1.31	19.02.2024	1h	3h	3	Verbindung von Frontend und backend für Artikel/Bestellungen	EG, KR, MD
1.32	20.02.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.33	23.02.2024	3h	-	1	Fehlerbehebung	EG
1.34	23.02.2024	2h	6h	3	Asynchrones Laden von Items beheben	EG, KR, MD
1.35	23.02.2024	2h	6h	3	Kunden-CRUD-Logik Fehlerbehebung	EG, KR, MD
1.36	24.02.2024	1h45	-	1	Refactor, Wechsel zu Formular-Komponenten für Ansichten	EG
1.36	24.02.2024	5h30	-	1	UI Design Customer und Articles	MD
1.37	26.02.2024	3h	-	1	Entwicklung der Rechnungsansicht	EG
1.38	26.02.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.38	29.02.2024	5h	-	1	UI, Errormessages, DB connection	MD
1.39	02.03.2024	1h30	3h	2	Entwicklung der Rechnungsfilterfunktion	EG, KR
1.40	03.03.2024	2h	-	2	Weiterentwicklung der Rechnungsfilterfunktion	EG, KR
1.41	03.03.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.41	03.03.2024	5h	-	1	UI logic, TreeView	MD
1.42	04.03.2024	1h	-	1	Weiterentwicklung der Rechnungsfilterfunktion	EG
1.43	05.03.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.44	06.03.2024	3h	6h	2	Integrationstests	EG, KR
1.45	07.03.2024	1h30	-	1	Frontend-Backend Verbindung optimieren	KR
1.46	08.03.2024	2h	-	1	UI-Tests durchführen	KR

1.47	09.03.2024	2h	6h	3	Code-Review Session	EG, KR, MD
1.48	10.03.2024	1h	3h	3	Scrum-Meeting	EG, KR, MD
1.49	11.03.2024	2h	-	1	Datenvalidierung in der Datenbank	MD,EG
1.49	11.03.2024	5h	-	1	Searchfilters, Buttons, Logic überarbeiten	MD
1.50	12.03.2024	2h	-	1	Leistungsverbesserungen im Backend	KR
1.51	13.03.2024	5h	-	1	Verbesserung der Dokumentqualität	KR
1.52	14.03.2024	4h	-	1	Rechnungsverwaltung verbessern	EG
1.53	15.03.2024	8h	24h	3	Projektabschluss	EG, KR, MD
Totaler Aufwand [h] 15.03.2024 246.75h						

Tabelle 1 Arbeitsjournal

Elias Götte = EG, Khabat Rammo = KR, Maximilian Degen = MD

Organisation

Projektmitglieder

Rolle	Name	Kürzel
Entwickler & Projektleitung	Maximilian Degen	MD
Entwickler	Khabat Rammo Elias Götte	KR EG

Tabelle 2 Projektmitglieder

Allgemein

Kommunikationsmittel	Outlook (Termine für Gruppenarbeiten und Abstimmungen) WhatsApp-Gruppenchat (Kleinere Abstimmungen) Teams Konferenzen (virtuelle Besprechungen)
Projekt Planungs-Dokument	PowerPoint über Office 365 Cloud geteilt. In diesem Dokument werden Informationen zur Planung (Zeitplan, nächste Termine und Schritte) geführt, sodass an Team-Meetings per Bildschirmübertragung auch visuelle Abstimmung stattfinden kann.

	<p>Geteilter Link</p> <p>Link wurde von khabat.rammo.ch@zbw-online.ch geteilt mit: eliaspgoette@gmail.com@zbw-online.ch maximiliandegen@hotmail.com@zbw-online.ch</p> <p>Teils Versionen des werden auch auf GitHub hochgeladen. (z.B. Tests, Abgabe relevante Dokumente Branch «Abgabe»)</p>
Projekt Dokument	<p>Word-Dokument über Office 365 Cloud geteilt.</p> <p>Geteilter Link</p> <p>Link wurde analog Planungsdokument ge- teilt</p>
Entwicklungs- und Abgabe relevante Dateien	<p>Für die Zusammenarbeit an entwicklungs- bezogene Dateien (.cs, etc.) wird das Git Repository verwendet.</p> <p>Git erstellt und berechtigt durch Elias Götte: eliasgoette/business-application-project (github.com)</p>

Tabelle 3 Allgemein

Zeitplan

Ein Zeitplan inkl. Ressourcen und Abwesenheiten werden stets mittels PowerPoint-Grafik von Projektleitung nachgetragen und in Team-Meetings besprochen. Hier ein Beispiel vom Stand 18.02.2024:

Zeitplan – nach Semesterplan

KW FACH NACH PRIO	5	6	7	8	9	10	11	12
65% Arbeit ELIAS Mo. & Di Nachmittag frei							ABGABE SONNTAG, 16.03.2024 23.59 Uhr	MODULPRÜFUNGEN
60% Arbeit MÄX Mo. (Morgen) & Di. (Nachmittag) Halbtags frei	ABWESEND AM WOCHENENDE							
80% Arbeit KHABAT Mo. & Fr. Nachmittag frei		ABWESEND						
	Planung	Arbeit				Abschluss		

Tabelle 4 Zeitplan

Aufgabenaufteilung

- Besprechungen via Team-Meetings
- Festgehalten in «Projekt Planungs-Dokument»
- Management und Überwachung aller Prozesse über DevOps

Einführung

Dieses Dokument dient als umfassende Dokumentation für das Projekt im Bereich der Auftragsverwaltung. Das Hauptziel dieses Projekts ist die Entwicklung einer effizienten und benutzerfreundlichen Softwarelösung zur Verwaltung von Aufträgen. Unser Fokus liegt dabei auf der Implementierung eines robusten Systems zur Datenverwaltung und -verarbeitung, wobei wir die etablierten Best Practices im Bereich Softwareengineering berücksichtigen.

Das Projekt wurde unter Verwendung moderner Softwareentwicklungsmethoden und -Werkzeuge durchgeführt. Es umfasst verschiedene Phasen von der Konzeption bis zur Implementierung, einschliesslich Entwurf, Entwicklung und Testing. Die Dokumentation bietet einen detaillierten Überblick über die angewandten Methoden, die Architektur des Systems und die genutzten Technologien.

Unser Ansatz gewährleistet, dass die resultierende Softwarelösung nicht nur funktional und effizient ist, sondern auch den Bedürfnissen und Erwartungen der Nutzer entspricht. Ein besonderer Fokus liegt auf der Qualität und Benutzerfreundlichkeit der Anwendung, um eine zuverlässige und zukunftsorientierte Lösung zu schaffen.

Methodik

In diesem Projekt haben wir uns für einen agilen Softwareentwicklungsansatz entschieden, wobei insbesondere die Praktiken und Prinzipien von Scrum und DevOps zur Anwendung kommen.

Scrum

Scrum ist als Rahmenwerk für unser Projektmanagement implementiert worden: [1]

Sprints

Die Entwicklung wurde in Sprints organisiert, wobei jeder Sprint auf 2 Woche festgelegt wurde.

Wöchentliche Stand-up-Meetings

Jeden Montag fand ein kurzes Stand-up-Meeting statt, um Fortschritte zu teilen, Herausforderungen zu identifizieren und die Aufgaben für die Woche zu koordinieren.

Sprint-Planung

Vor jedem Sprint haben wir eine Sprint-Planungssitzung durchgeführt, um Aufgaben zu priorisieren und Ziele festzulegen.

Sprint-Review

Am Ende jedes Sprints wurde ein Review-Meeting abgehalten, um die geleistete Arbeit zu demonstrieren und Feedback zu sammeln.

Retrospektiven

Um unseren Prozess kontinuierlich zu verbessern, führten wir nach jedem Sprint eine Retrospektive durch, in der wir reflektierten, was gut lief und was verbessert werden könnte.

DevOps

DevOps-Praktiken wurden eingeführt, um die Zusammenarbeit zwischen Entwicklung und Betrieb zu stärken und die Softwarequalität durch kontinuierliche Integration und Tests zu verbessern. Hierzu zählt auch die Verwendung von Azure DevOps für die Projektverwaltung, welche folgende Vorteile mit sich bringt: [2]

Board zur Aufgabenverwaltung

Wir nutzten das Azure DevOps Board, um unsere Arbeit zu organisieren, User Stories zu verwalten und den Fortschritt jedes Sprints zu verfolgen.

Repos für Versionskontrolle

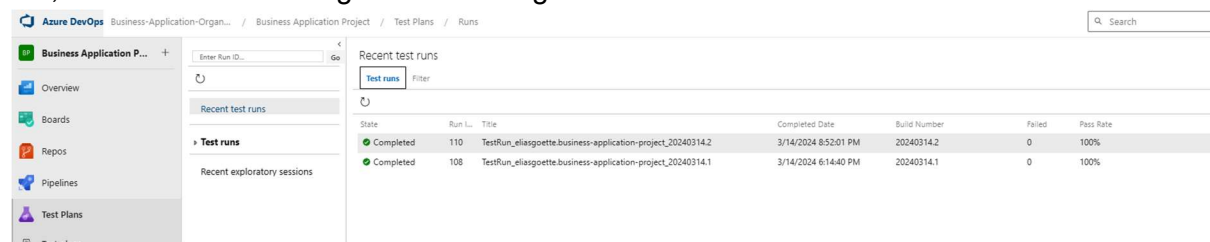
Der Code wurde in Azure Repos hochgeladen, wodurch das Team den Überblick über Änderungen behalten und effizient zusammenarbeiten konnte.

Pipelines für Build und Release

Azure Pipelines wurden verwendet, um automatisierte Builds und Tests durchzuführen und sicherzustellen, dass Änderungen nicht zu Regressionen im System führten.

Testpläne

Um die Qualität zu gewährleisten, wurden Testpläne erstellt und ausgeführt, um sicherzustellen, dass alle Anforderungen korrekt umgesetzt wurden.



State	Run ID	Title	Completed Date	Build Number	Failed	Pass Rate
Completed	110	TestRun_eliasgötte.business-application-project_20240314.2	3/14/2024 8:52:01 PM	20240314.2	0	100%
Completed	108	TestRun_eliasgötte.business-application-project_20240314.1	3/14/2024 6:14:40 PM	20240314.1	0	100%

Abbildung 1 Test run

Diese Methoden und Werkzeuge haben es uns ermöglicht, effizient als Team zu arbeiten und ein qualitativ hochwertiges Produkt für unser Projekt zu entwickeln.

GitHub

Versionskontrolle mit GitHub

Die Nutzung von GitHub als Plattform für die Versionskontrolle spielt eine zentrale Rolle in unserem Projekt. Durch die verteilte Versionskontrollfunktion von Git, gepaart mit den benutzerfreundlichen Oberflächen und Werkzeugen von GitHub, konnte unser Team den Quellcode effizient verwalten und gemeinsam an der Codebasis arbeiten. Die Möglichkeit, jederzeit zu früheren Versionen zurückzukehren und parallele Entwicklungszweige (Branches) zu pflegen, erhöhte unsere Produktivität und verringerte das Risiko von Konflikten im Code.

Code-Reviews und Zusammenarbeit

GitHub erleichtert nicht nur das gemeinsame Coden, sondern fördert auch die Qualitätssicherung durch Pull Requests und Code-Reviews. Jede bedeutende Änderung wurde als Pull Request eingereicht, über den das Team diskutieren, Verbesserungsvorschläge einbringen und letztlich die Änderungen in die Hauptentwicklungslinie (Master Branch) integrieren konnte. Dieser Prozess stellte sicher, dass jeder Codebeitrag den Projektstandards entsprach und von mindestens einem anderen Teammitglied überprüft wurde.

Integration mit Azure DevOps

Obwohl wir Azure DevOps für das Sprint- und Task-Management verwendet haben, bot GitHub eine zusätzliche Ebene der Übersichtlichkeit. Die Verknüpfung von Commits und Branches mit Azure DevOps Work Items ermöglichte eine nahtlose Integration der Code-Entwicklung in unseren agilen Prozess. Dies stellte sicher, dass die Codebasis und das Projektmanagement synchron blieben.

Fazit

Insgesamt bot GitHub unserem Team ein robustes System für das Quellcode-Management, das entscheidend zur Qualität und Kollaboration beitrug. Die Kombination aus GitHub und Azure DevOps schuf ein agiles, reaktives und qualitätsorientiertes Arbeitsumfeld, das die erfolgreiche Umsetzung unseres Schulprojekts massgeblich unterstützte.

Agile Durchführung und Sprint Management

Im Zuge der agilen Durchführung unseres Projekts haben wir Sprints sorgfältig geplant und durchgeführt. Jeder Sprint begann mit einem klar definierten Sprint Backlog, das aus spezifischen User Stories bestand, die für den jeweiligen Sprint geplant waren.

Sprint Backlog und Task Breakdown

Für jeden Sprint wurde ein detailliertes Backlog erstellt, das die geplanten User Stories enthielt. Diese wurden anschliessend in geeignete Tasks aufgeteilt, die innerhalb des Sprints bearbeitet wurden, um die Ziele des Sprints zu erreichen.

Sprint Review und Retrospektive

Am Ende jedes Sprints haben wir ein Sprint Review sowie eine Retrospektive durchgeführt. Diese Sitzungen waren entscheidend, um die geleistete Arbeit zu bewerten und Verbesserungsmöglichkeiten für den nächsten Sprint zu identifizieren.

The screenshot shows a Jira ticket interface. At the top, it says 'TASK 35' and '35 Sprint review and retrospective'. Below this, it indicates '1 comment' and 'Add tag'. The ticket is assigned to 'Khabat Rammo' and has a status of 'Done'. The area is 'Business Application Project' and the iteration is 'Business Application Project/Sprint 0'. A comment by 'Elias Götte' dated 'Feb 4 (edited)' is visible. The comment content is as follows:

Project Setup Retrospective

Completed Tasks:

- GitHub Setup: We established a robust version control system, which will be crucial for managing future code changes.
- Azure Work Items & Pipeline: The completion of Azure work items and pipeline setup has laid the groundwork for a solid CI/CD process.
- VS Windows Forms and MS Test Suite Project: Initiating the main application and test suite early on has ensured that we are building on a test-driven approach.
- Architecture and DB Design Discussions: These discussions have provided a clear path forward for the development and should mitigate potential architectural issues.

Insights:

- The early focus on key infrastructure tasks has set a strong foundation for the project.
- The team's ability to complete these foundational tasks on time reflects effective planning and execution.
- Successful discussions around architecture and database design indicate good team collaboration and decision-making.

Reflection:

- The completed tasks show a strong start to the project. However, it's important to reflect on any challenges faced during these tasks to improve our approach in future phases.
- Identifying what worked well and what didn't is essential for continuous improvement.

Project Setup Review

In-Progress Tasks:

- Documentation Structure & Planning: These tasks are crucial for long-term maintenance and user support, indicating foresight in the project's lifecycle.
- Documentation Standards Guidelines: By establishing these guidelines, the team will ensure consistency and quality in documentation.
- GitHub Tag Creation: This will enhance our ability to manage releases and track issues effectively.

Next Steps:

- Complete In-Progress Tasks: These tasks need to be prioritized to prevent becoming blockers.
- Review Progress: Regular check-ins on the progress of in-progress tasks will help in maintaining the timeline.
- Anticipate Future Tasks: Planning for upcoming tasks now can save time later and help maintain project momentum.
- Process Evaluation: Taking time to evaluate our current workflows can reveal opportunities for improvement.

Recommendations:

- Allocate dedicated time for the team to focus on completing the documentation-related tasks.
- Continue to engage the team in regular retrospectives to ensure that everyone has a chance to voice their opinions on the project's direction and methodology.
- Keep the team motivated by celebrating completed tasks and maintaining transparency on project progress.

Abbildung 2 Sprint Review

Sprint 0: Projekteinrichtung

Abschnitt	Beschreibung
Abgeschlossene Aufgaben	- GitHub-Einrichtung für Versionskontrolle- Azure Work Items & Pipeline Setup- Initiation von VS Windows Forms und MS Test Suite Projekt- Architektur- und DB-Design-Diskussionen
Erkenntnisse	- Starke Grundlage durch Fokus auf Infrastruktur- Effektive Planung und Ausführung des Teams- Gute Teamkollaboration bei der Architektur

Reflexion	- Überprüfung der Herausforderungen zur Verbesserung in zukünftigen Phasen- Identifikation von Stärken und Schwächen für kontinuierliche Verbesserung
In Bearbeitung befindliche Aufgaben	- Dokumentationsstruktur & Planung- Richtlinien für Dokumentationsstandards- Erstellung von GitHub-Tags - Sprint Review dokumentieren
Nächste Schritte	- Priorisierung und Abschluss offener Aufgaben- Regelmässige Überprüfung des Fortschritts- Zukünftige Aufgaben planen- Prozessbewertung für Verbesserungsmöglichkeiten
Empfehlungen	- Dedizierte Zeit für dokumentationsbezogene Aufgaben- Regelmässige Retrospektiven zur Teamkollaboration- Motivation und Transparenz im Team aufrechterhalten

Tabelle 5 Sprint 0

Sprint 1: Ausführung und Überprüfung

Abschnitt	Beschreibung
Abgeschlossene Aufgaben	- DB-Migration für Hierarchical Article Group Management- Modellerstellung für Invoice Creation
Erkenntnisse	- Die Fertigstellung kritischer Datenbankaufgaben trägt zur robusten Grundlage des Projekts bei - Effektive Zuweisung und Abschluss von Datenmodellierungsaufgaben
In Bearbeitung befindliche Aufgaben	- Dokumentation für Hierarchical Article Group Management - UI-Entwicklung für CRUD Customers und Invoice Creation
Anstehende Aufgaben	- Daten-Seedings zu Migrationen hinzufügen - UI für Invoice Creation entwickeln - Sprint Review dokumentieren
Reflexion	- Die Notwendigkeit einer detaillierten Planung für UI-bezogene Aufgaben wurde identifiziert, um Verzögerungen zu vermeiden - Wichtigkeit der Dokumentation als fortlaufender Prozess betont
Nächste Schritte	- Priorisierung der UI-Entwicklung, um das Benutzererlebnis zu verbessern - Kontinuierliche Dokumentation der Entwicklungsfortschritte- Vorbereitung auf die Sprint-Review-Sitzung
Empfehlungen	- Dokumentation parallel zur Entwicklung führen, um den Überblick zu behalten

Tabelle 6 Sprint 01

Sprint 2: Details zu Aufgaben und Fortschritt

Abschnitt	Beschreibung
Abgeschlossene Aufgaben	- Erstellung der DB-Migration mit temporalen Features
In Bearbeitung befindliche Aufgaben	- Verbindung von Frontend und Backend für CRUD Articles/Orders - Auswahl für Gridview
Anstehende Aufgaben	- Erstellung einer Rechnungsaufstellung mit historischer Adressgenauigkeit und Benutzerfilterung - Sprint Review dokumentieren
Reflexion	- Die Notwendigkeit für eine sorgfältige Planung bei der Integration von Frontend und Backend wurde deutlich
Nächste Schritte	- Abschluss der in Bearbeitung befindlichen Aufgaben - Beginn der Arbeiten an der Rechnungsaufstellung
Empfehlungen	-

Tabelle 7 Sprint 02

Sprint 3: Aufgabenstatus und Handlungsbedarf

Abschnitt	Beschreibung
Abgeschlossene Aufgaben	- Fehlerbehebung: Synchrones Laden von Elementen, das Fehler verursacht
In Bearbeitung befindliche Aufgaben	- Erstellung der Logik für die Benutzeroberfläche (CRUD Customers) - Erstellung eines generischen Controllers
Anstehende Aufgaben	- Fehlerbehebung: Erstellung von Elementen mit Navigationsattributen verursacht Fehler - Sprint Review dokumentieren
Reflexion	- Die schnelle Identifikation und Behebung von Synchronisierungsfehlern zeigt die Reaktionsfähigkeit des Teams
Nächste Schritte	- Abschluss der in Bearbeitung befindlichen Entwicklungsaufgaben für die Benutzeroberfläche - Untersuchung und Behebung von Fehlern bei der Erstellung von Elementen mit Navigationsattributen
Empfehlungen	- Weitere Prüfung asynchroner Vorgänge im Code zur Vermeidung zukünftiger Fehler - Konsistente Dokumentation von Bugfixes und deren Auswirkungen auf die Anwendung

Tabelle 8 Sprint 03

Sprint 4: Überprüfung und Fortschritt

Abschnitt	Beschreibung
Abgeschlossene Aufgaben	Keine abgeschlossenen Aufgaben verzeichnet.
In Bearbeitung befindliche Aufgaben	- Erstellung der Logik für die Benutzeroberfläche (CRUD Customers)
Anstehende Aufgaben	- Sprint Review dokumentieren
Reflexion	- Notwendigkeit einer gründlichen Analyse, warum keine Aufgaben abgeschlossen wurden, und Bewertung der aktuellen Hindernisse und Herausforderungen.
Nächste Schritte	- Abschluss der begonnenen Aufgaben - Vorbereitung auf das Sprint-Review, um die Herausforderungen des Sprints zu diskutieren und zu verstehen.
Empfehlungen	- Identifizierung von Blockaden und Entwicklung von Lösungsstrategien - Anpassung der Aufgabenpriorisierung und Ressourcenallokation

Tabelle 9 Sprint 04

Sprint 5: Planung und Erwartungen

Abschnitt	Beschreibung
Abgeschlossene Aufgaben	- Entwicklung der Logik für Filter - Entwicklung der Datenpräsentation

In Bearbeitung befindliche Aufgaben	- UI-Design für Invoice Listing with Historical Address Accuracy and User Filters
Anstehende Aufgaben	- Review und Reflexion des Sprints
Reflexion	- Es ist wichtig zu analysieren, warum keine Aufgaben in Sprint 4 abgeschlossen wurden und Massnahmen zu ergreifen, um die Produktivität in Sprint 5 zu steigern.
Nächste Schritte	- Fertigstellung des UI-Designs für die Rechnungsaufstellung - Vorbereitung und Durchführung des Sprint-Reviews
Empfehlungen	- Bewertung der Zeitplanung und Ressourcenverteilung, um Engpässe zu identifizieren und zukünftig zu vermeiden - Fortlaufende Überwachung des Fortschritts um sicherzustellen, dass alle Aufgaben rechtzeitig abgeschlossen werden können

Tabelle 10 Sprint 05

Initialisierung Azure DevOps [3]

Schätzung des Produkt-Backlogs

Für die Schätzung unseres Produkt-Backlogs haben wir die **T-Shirt-Grössen-Methode** [4] angewandt, eine gängige Technik in agilen Projekten, um den relativen Aufwand von User Stories zu bewerten. Diese Methode ist besonders hilfreich, um ein gemeinsames Verständnis von Aufwand und Komplexität innerhalb des Teams zu schaffen.

Durchführung der Schätzung:

User Stories Sammlung:

Zunächst haben wir alle User Stories für die funktionalen Anforderungen zusammengetragen. [5]

Einführung in die Methode

Um ein einheitliches Verständnis der Bewertungskriterien sicherzustellen, wurde jedes Teammitglied gründlich mit der T-Shirt-Grössen-Schätzmethode vertraut gemacht. Diese Methode verwendet unterschiedliche Grössenbezeichnungen (XS, S, M, L, XL), die jeweils für einen zunehmenden Grad an Aufwand und Komplexität der User Stories stehen. Wir haben spezifische Richtlinien definiert, was jede Grösse repräsentiert, um Konsistenz in unseren Schätzungen zu gewährleisten. Dabei haben wir berücksichtigt, dass die Schätzungen weniger auf exakten Messungen, sondern vielmehr auf dem relativen Vergleich und der Erfahrung des Teams basieren.

Diskussion der User Stories

Wir diskutierten jede User Story, um sicherzustellen, dass alle Mitglieder den Umfang und die Anforderungen verstanden haben.

Abstimmung über WhatsApp

Für die Schätzung haben wir eine WhatsApp-Umfrage erstellt, bei der jedes Teammitglied abstimmen konnte, welche T-Shirt-Grösse am besten zu jeder User Story passt.

Auswertung der Ergebnisse

Die Analyse der Abstimmungsergebnisse ermöglichte uns, ein klares Bild der am häufigsten gewählten T-Shirt-Größen für jede User Story zu erhalten. Die Größe, die die Mehrheit der Stimmen für eine bestimmte User Story erhielt, wurde als endgültige Schätzung festgelegt. Bei Abstimmungsergebnissen, die keine klare Mehrheit aufwiesen, führten wir eine eingehende Diskussion, um zu einem Konsens zu gelangen. Dieser Ansatz gewährleistete, dass alle Teammitglieder in den Entscheidungsprozess einbezogen wurden und dass die endgültige Schätzung die kollektive Einschätzung des Teams widerspiegelt.

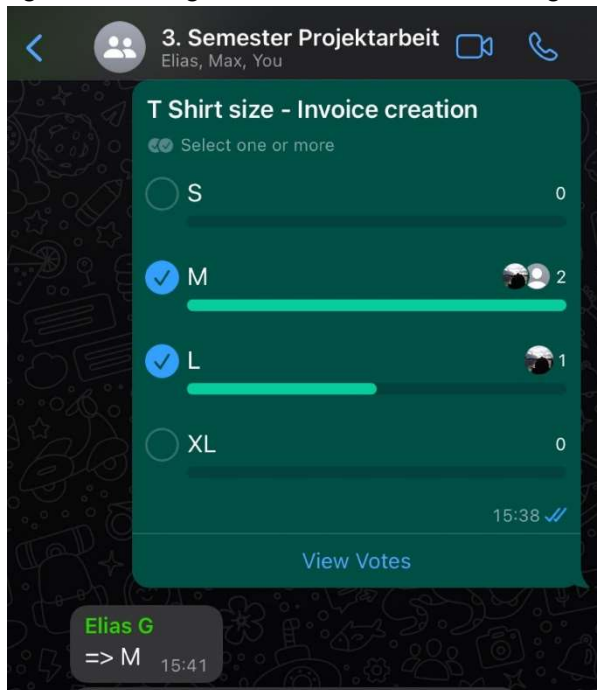


Abbildung 4 Schätzung des Produkt-Backlogs

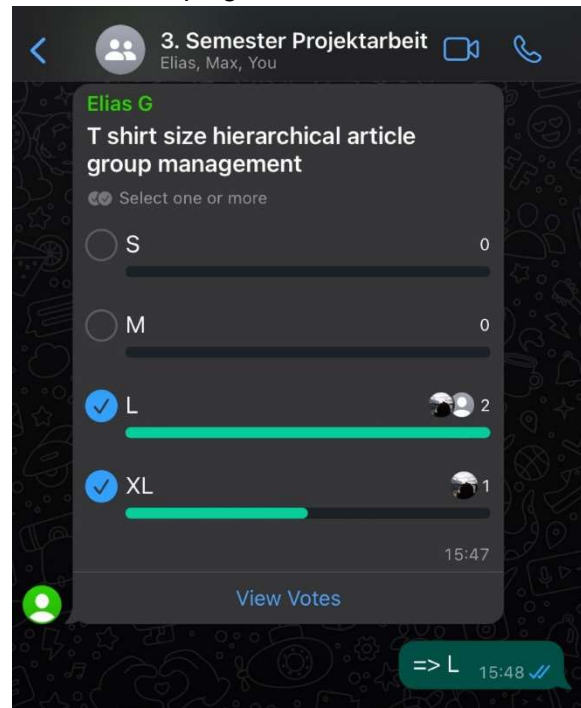


Abbildung 3 Schätzung des Produkt-Backlogs

Entscheidungsfindung

Die Ergebnisse der Umfrage haben gezeigt, dass das Team unterschiedliche Auffassungen über den Aufwand einiger User Stories hatte.

Rationale für Schätzwerte

Die Schätzwerte wurden auf Basis von Erfahrungswerten, Komplexität der Anforderungen, und der geschätzten Zeit zur Umsetzung festgelegt. Durch die Methode der T-Shirt-Größen konnten wir eine gemeinsame Sprache für die Aufwandsschätzung entwickeln und eine Einigkeit im Team erreichen.

Diese Vorgehensweise ermöglicht es uns, eine transparente und nachvollziehbare Schätzung für unser Backlog zu erstellen, die allen Teammitgliedern klar kommuniziert werden kann.

Git-Management und Sourcecode-Verwaltung

In unserem Projekt kam Git als Versionskontrollsystem zum Einsatz. Für eine effiziente Sourcecode-Verwaltung wurde ein GitHub-Account erstellt und ein Repository eingerichtet, das mit unserem Azure DevOps-Projekt verknüpft wurde.

Branching und Merging Strategie

Um die Entwicklung produktiv und organisiert zu gestalten, haben wir ein durchdachtes Branching- und Merging-Konzept entwickelt und konsequent angewendet. Dieses Konzept war entscheidend für die Strukturierung unserer Arbeit und das Management von Features, Bugfixes und Releases. [6]

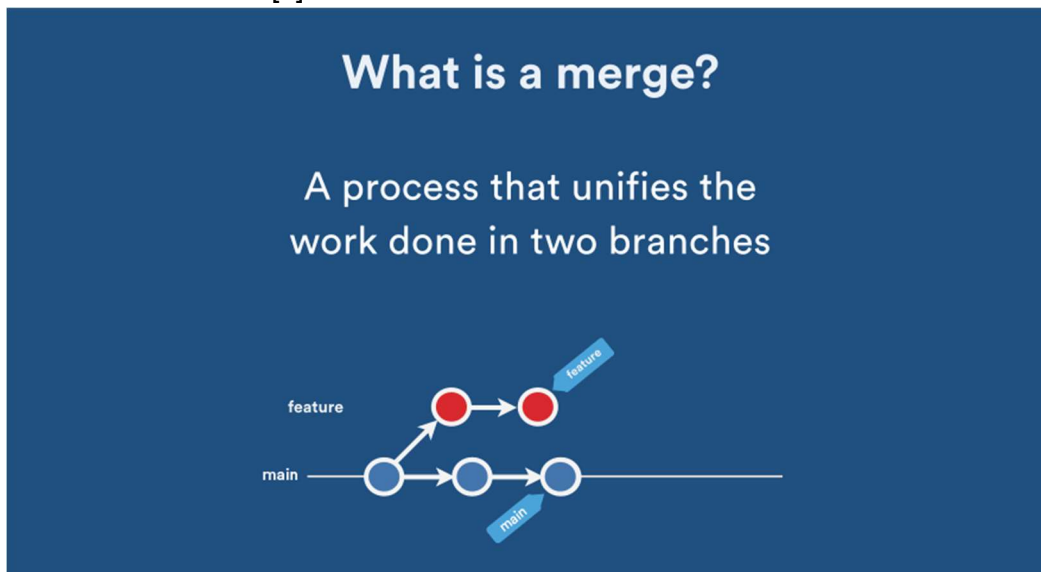


Abbildung 5 Branching und Merging Strategie

Commit Konvention und Tags

Im Laufe des Projekts wurden regelmässig zweckmässige Commits mit aussagekräftigen Nachrichten erstellt, um den Überblick über die Entwicklungsschritte zu bewahren. Zusätzlich wurden Tags gemäss den Projektmeilensteinen gesetzt, um wichtige Versionen und Zustände des Codes zu markieren. [7]

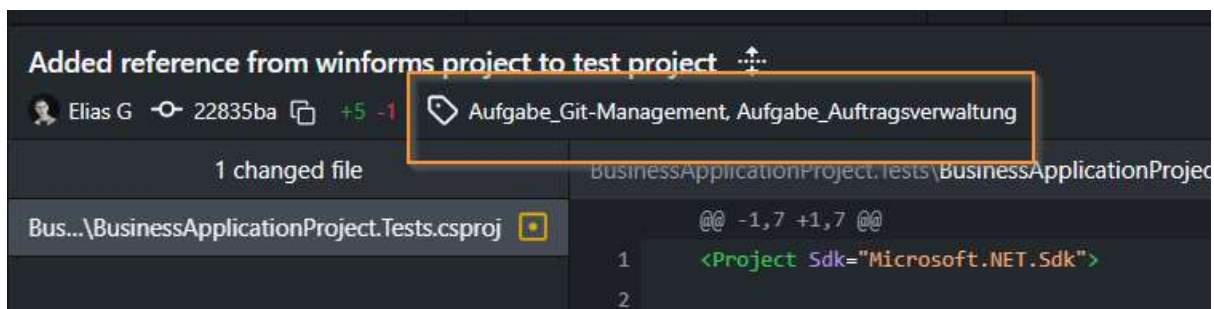


Abbildung 6 Tags

Integration in die Projektumgebung

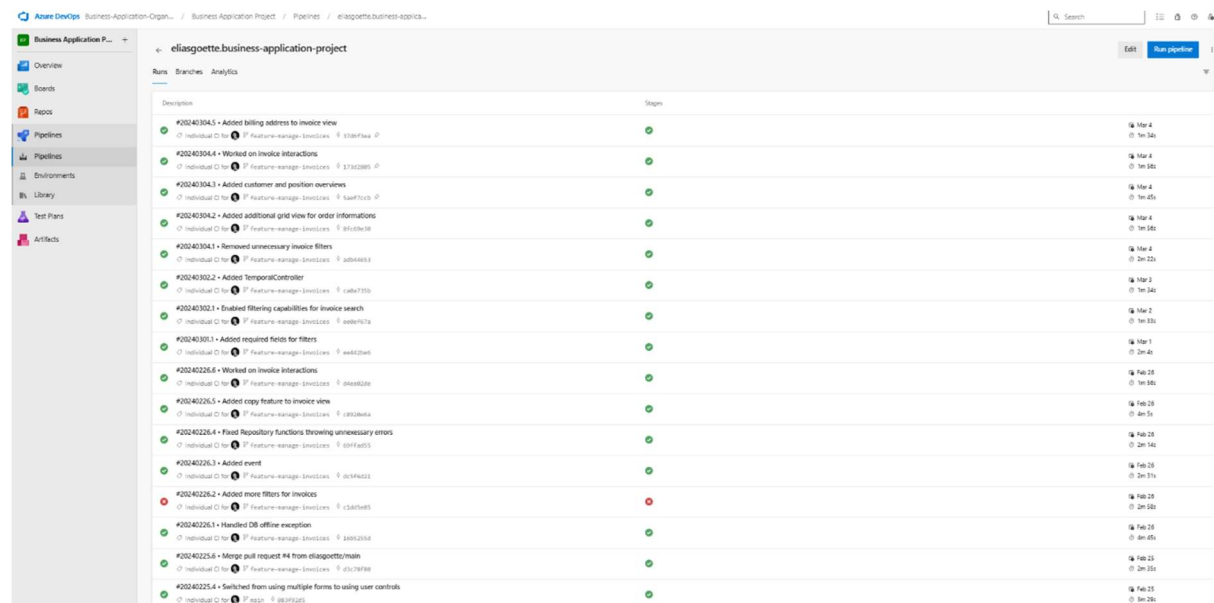
Die Integration von Git mit Azure DevOps ermöglichte es uns, eine kollaborative Umgebung für das Code-Management zu schaffen, die automatisierte Builds und Tests sowie eine klare Nachvollziehbarkeit von Änderungen unterstützt. Des Weiteren wurde unser Dozent als Contributor zum Git-Repository hinzugefügt, um die Transparenz und Zusammenarbeit zu fördern

Continuous Integration und Automatisierte Tests

Im Rahmen unseres Entwicklungsprozesses haben wir eine Release-Pipeline in Azure DevOps eingerichtet, die eine Schlüsselrolle in unserer CI/CD-Strategie spielt. [8]

Aufbau der Release-Pipeline

Die konfigurierte Release-Pipeline wurde so eingerichtet, dass bei jedem Commit im Repository automatisch ein Build des gesamten Projekts ausgelöst wird. Zudem führt sie alle vorhandenen Unittests durch, um sicherzustellen, dass Änderungen keine bestehenden Funktionalitäten beeinträchtigen. [8]



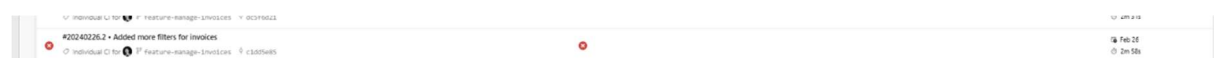
The screenshot shows the Azure DevOps Pipelines interface for the project 'eliasgoette.business-application-project'. The left sidebar contains navigation links for Overview, Boards, Repos, Pipelines, Environments, Library, Test Plans, and Artifacts. The main area displays a table of pipeline runs with columns for Description, Status, and Origin. The table lists 20 runs, most of which are successful (green checkmark), but one run (#20240226.2) is failed (red X).

Description	Status	Origin
#20240104.5 • Added billing address to invoice view	Success	Man 4 (1m 34s)
#20240204.4 • Worked on invoice interactions	Success	Man 4 (1m 34s)
#20240104.3 • Added customer and position overviews	Success	Man 4 (1m 45s)
#20240104.2 • Added additional grid view for order informations	Success	Man 4 (1m 34s)
#20240104.1 • Removed unnecessary invoice filters	Success	Man 4 (2m 22s)
#20240102.2 • Added TemporalController	Success	Man 2 (1m 34s)
#20240102.1 • Enabled filtering capabilities for invoice search	Success	Man 2 (1m 33s)
#20240101.1 • Added required fields for filters	Success	Man 1 (2m 4s)
#20240226.6 • Worked on invoice interactions	Success	Feb 26 (1m 34s)
#20240226.5 • Added copy feature to invoice view	Success	Feb 26 (2m 3s)
#20240226.4 • Fixed Repository functions throwing unnecessary errors	Success	Feb 26 (2m 14s)
#20240226.3 • Added event	Success	Feb 26 (2m 11s)
#20240226.2 • Added more filters for invoices	Failed	Feb 26 (2m 58s)
#20240226.1 • Handled DB offline exception	Success	Feb 26 (2m 45s)
#20240225.6 • Merge pull request #4 from eliasgoette/main	Success	Feb 25 (2m 35s)
#20240225.4 • Switched from using multiple forms to using user controls	Success	Feb 25 (2m 26s)

Abbildung 7 Pipeline

Feedback-Mechanismen und Fehlerbehandlung

Im Falle eines fehlgeschlagenen Builds erhält der verantwortliche Entwickler, der den Commit durchgeführt hat, automatisch eine Fehlermeldung per E-Mail. Dies stellt sicher, dass Probleme schnell erkannt und behoben werden können. [8]



The screenshot shows a failed pipeline run in the Azure DevOps Pipelines interface. The run is identified as '#20240226.2 • Added more filters for invoices' and is marked with a red 'X' icon, indicating a failure. The status bar at the bottom shows 'Failed' and '2m 58s'.

Description	Status	Origin
#20240226.2 • Added more filters for invoices	Failed	Feb 26 (2m 58s)

Abbildung 8 Feedback-Mechanismen

Bedeutung der CI für das Projekt

Durch die Implementierung von Continuous Integration konnten wir die Stabilität des Codes erhöhen, Fehler frühzeitig erkennen und die Entwicklungszyklen verkürzen. Dies war ein wesentlicher Bestandteil unserer Strategie, um eine hohe Qualität unserer Softwarelösung sicherzustellen.

Datenbank- und Anwendungsfunktionen

Unsere Datenbankarchitektur wurde mit Blick auf Effizienz und Skalierbarkeit entworfen. Die Implementierung einer hierarchischen Artikelgruppenverwaltung erforderte die Verwendung von rekursiven Common Table Expressions (CTE), um verschachtelte Beziehungen darzustellen. Für das Rechnungsmanagement setzten wir Window-Funktionen für Jahresvergleiche ein, was die Erweiterung unseres Datenmodells für den effektiven Umgang mit temporalen Daten nach sich zog. Diese Funktionen verbesserten nicht nur nachweislich die Funktionalität der Anwendung, sondern demonstrierten auch unsere technische Kompetenz.

Datenbankmodell

Das Datenbankmodell für unser Auftragsverwaltungssystem beinhaltet sieben Kernentitäten, die zusammen das Rückgrat der Anwendung bilden. Nachfolgend finden Sie eine detaillierte Beschreibung der Entitäten und ihrer Beziehungen zueinander:

1. Address (Adresse)

Jede Adresse kann mehreren Kunden zugeordnet sein.

Felder: Id, Country, ZipCode, City, StreetAddress

2. Customer (Kunde)

Jeder Kunde ist genau einer Adresse zugeordnet.

Ein Kunde kann mehrere Bestellungen aufgeben.

Felder: Id, CustomerNumber, FirstName, LastName, Email, Website, PasswordHash

3. ArticleGroup (Artikelgruppe)

Artikelgruppen können eine hierarchische Struktur haben

Felder: Id, Name

4. Article (Artikel)

Jeder Artikel gehört zu genau einer Artikelgruppe.

Ein Artikel kann in mehreren Bestellungen vorkommen.

Felder: Id, ArticleNumber, Name, Price

5. Order (Bestellung)

Jede Bestellung ist genau einem Kunden zugeordnet.

Jede Bestellung besteht aus mehreren Positionen.

Jeder Bestellung ist genau eine Rechnung zugeordnet.

Felder: Id, OrderNumber, Date

6. Position

Eine Position verbindet eine Bestellung mit einem Artikel.

Felder: Id, PositionNumber, Quantity

7. Invoice (Rechnung)

Jede Rechnung ist genau einer Bestellung zugeordnet.

Felder: Id, InvoiceNumber, DueDate, Discount, TaxPercentage, PaymentMethod, PaymentStatus

Analyse der Commit-Historie

Unsere Commit-Historie spiegelt eine disziplinierte Versionierung und einen iterativen Entwicklungsprozess wider. Jeder Commit repräsentierte einen substantiellen Fortschritt in

unserem Projekt. Wir nutzten Tags gezielt, um bedeutsame Meilensteine zu markieren und sorgten für Transparenz bei allen Beteiligten, einschliesslich unseres Dozenten.

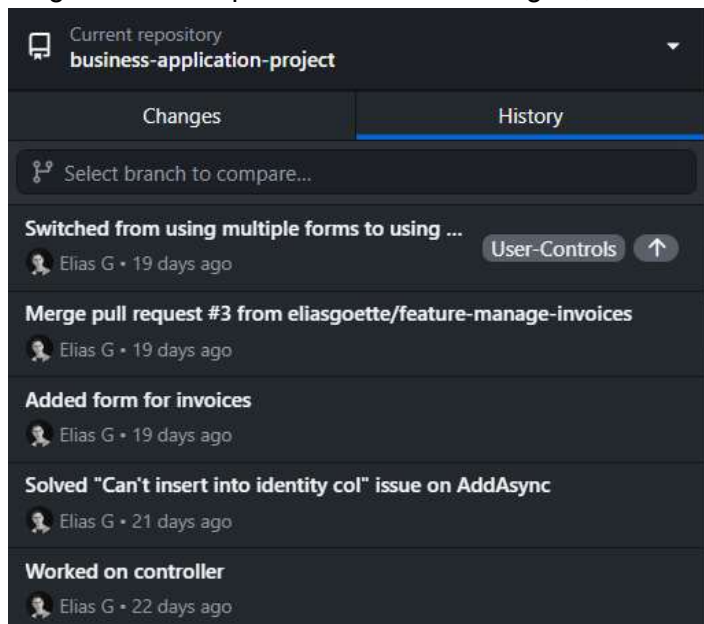


Abbildung 9 Commit-Historie

UI/UX

Die Entwicklung unserer Benutzeroberfläche war ein iterativer Prozess mit Nutzerfokus. Benutzerstudien und Prototyping führten zu mehreren Überarbeitungen, bis eine benutzerfreundliche und optisch ansprechende Lösung gefunden war. [9]

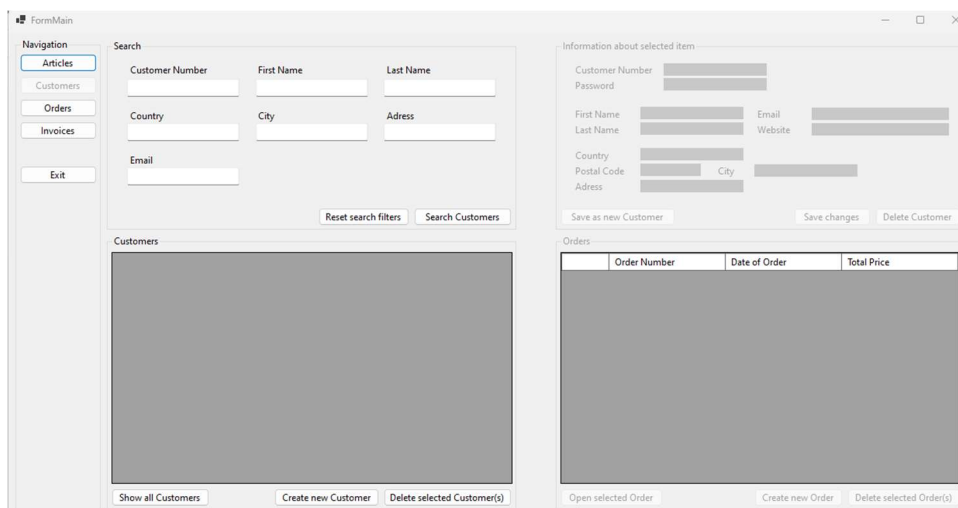


Abbildung 10 GUI

Inbetriebnahme der Applikation und Generierung der Datenbank

Die folgende Dokumentation beschreibt die erforderlichen Schritte, um die Anwendung zu starten und die Datenbank mit den notwendigen Testdaten zu generieren, die für die Überprüfung der funktionalen Anforderungen erforderlich sind.

Voraussetzungen:

- Installation von .NET Core SDK und Entity Framework Core Tools. [10]
- Eine lauffähige Instanz von Microsoft SQL Server oder ein kompatibler SQL-Dienst. [11]
- Das Projekt muss von GitHub geklont werden.

Schritte zur Einrichtung der Datenbank:

1. Konfigurieren der Datenbankverbindung:

Stellen Sie sicher, dass die Verbindungszeichenfolge in der **OnConfiguring-Methode** des **AppDbContext** korrekt auf Ihren Datenbankserver zeigt.

1. Migrationen Ausführen:

Nutzen Sie Entity Framework Core Migrations, um die Datenbankstruktur aufzubauen:

```
1. dotnet ef migrations add InitialCreate
2. dotnet ef database update
```

Diese Befehle erzeugen die Datenbanktabellen basierend auf den im DbContext definierten Modellen.

Generierung von Testdaten:

Daten Seeder:

Die **InsertSeedData-Methode** im DataSeeder wird aufgerufen, um Testdaten in die Datenbank einzufügen. Diese Methode wird während der Initialisierung der Datenbank in der **OnModelCreating-Methode** des AppDbContext ausgeführt.

Beispieldaten:

Es werden Beispieldaten für Adressen, Kunden, Artikelgruppen, Artikel, Bestellungen und Rechnungen erstellt und in die Datenbank eingefügt. Jedes Entity wird entsprechend den Beziehungen konfiguriert, die in der OnModelCreating-Methode definiert sind.

Starten der Applikation:

1. Applikationsstart:

Mit dem Befehl **dotnet run** im Stammverzeichnis des Projekts starten Sie die Applikation.

Wichtige Code-Komponenten:

a) Controller

Der generische Controller `Controller<T>` dient als intermediär zwischen dem Datenzugriff und der Präsentationslogik. Er bietet grundlegende CRUD-Operationen für alle Entities.

b) Repository:

Das `Repository<T>` abstrahiert die Datenzugriffslogik und stellt sicher, dass Operationen wie das Hinzufügen und Entfernen von Entities auf die Datenbank angewendet werden.

c) DbContext

Der `AppDbContext` definiert die `DbSets` für alle Entities und die Konfigurationen für ihre Beziehungen und Tabellen.