Hochschule Bremerhaven

Fachbereich II
Management und Informationssysteme
Wirtschaftsinformatik B.Sc.

Modul
Qualitätsmanagement

Semesteraufgabe

Entwicklung einer Hausverwaltung

Vorgelegt von:	Junior Lesage Ekane Njoh	MatNr. 40128
	Steve Aguiwo II	MatNr. 40088
	Franck Majeste Dogmo Silatsa	MatNr. 00000
Vorgelegt am:	5. März 2025	
Dozent:in:	Prof. Dr. Karin Vosseberg	

Inhaltsverzeichnis

1 Einleitung

Die Verwaltung von Gebäuden und deren Energieverbrauch stellt in der Praxis eine zentrale Herausforderung dar. Insbesondere in Mehrfamilienhäusern oder Wohnanlagen ist eine effiziente und übersichtliche Erfassung von Zählerständen erforderlich, um Verbrauchsdaten transparent zu machen und eine gerechte Abrechnung zu ermöglichen. Im Rahmen dieses Projekts entwickeln wir als Gruppe einen Prototyp für eine Hausverwaltungssoftware, die sich auf die digitale Erfassung, Verwaltung und Analyse von Zählerständen konzentriert.

Die Umsetzung erfolgt als webbasierte Anwendung mit einer intuitiven Benutzeroberfläche und einer zuverlässigen Datenverarbeitung. Unsere Hausverwaltung ermöglicht es, Gebäude, Zähler und Verbrauchsdaten zu verwalten, Zählerablesungen zu dokumentieren und historische Verbrauchswerte grafisch darzustellen. Dabei werden sowohl technische als auch organisatorische Aspekte berücksichtigt, um eine realitätsnahe und funktionale Lösung zu entwickeln.

Ein wesentlicher Bestandteil des Projekts ist das Review der Anforderungen sowie die Entwicklung eines fundierten Testkonzepts, um sicherzustellen, dass der Prototyp stabil, fehlerresistent und effizient arbeitet. Im Rahmen unserer Ausarbeitung dokumentieren wir die einzelnen Projektschritte detailliert und analysieren die gewonnenen Erkenntnisse. Unser Ziel ist es, ein praxisnahes und gut strukturiertes System zu entwerfen, das die wesentlichen Funktionen einer Hausverwaltung abbildet.

Die Entwicklung des Prototyps folgt einem iterativen Ansatz. Zu Beginn wurden die Anforderungen überprüft und überarbeitet, um Widersprüche oder Unklarheiten zu beseitigen. Anschließend wurden konkrete Testfälle definiert, um die Kernfunktionen zu validieren. Die Tests umfassen funktionale Prüfungen, negative Tests sowie Leistungstests, um sowohl die korrekte Funktionalität als auch die Systemgrenzen zu ermitteln. Schließlich wurde der Prototyp entsprechend der definierten Anforderungen und Testfälle umgesetzt und evaluiert.

Mit dieser Arbeit dokumentieren wir den gesamten Entwicklungsprozess, von der Anforderungsanalyse über die Testkonzeption bis hin zur Implementierung und Evaluation des Prototyps.

2 Anforderungsanalyse

2.1 Review der Anforderungen

Im Rahmen dieses Projekts haben wir ein `technisches Review` nach ISO 20246 durchgeführt. Diese Methode wurde gewählt, da sie eine frühe Fehlererkennung in der Anforderungsphase ermöglicht und sich besonders für dokumentenbasierte Analysen eignet.

Das Review-Team bestand aus allen drei Projektmitgliedern. Die Analyse erfolgte in zwei Schritten:

1. **Individuelle Prüfung:** Jedes Teammitglied hat alleine für sich die Anforderungen unabhängig nach definierten Kriterien überprüft.
2. **Gemeinsame Konsolidierung**: In einer Sitzung wurden die identifizierten Probleme besprochen und Verbesserungsvorschläge erarbeitet.

Die Überprüfung erfolgte anhand folgender Kriterien:

- `Vollständigkeit:` Sind alle relevanten Aspekte der Hausverwaltung abgedeckt?
- `Eindeutigkeit:` Sind die Anforderungen so formuliert, dass keine Missverständnisse entstehen?
- `Widerspruchsfreiheit:` Gibt es logische oder inhaltliche Widersprüche?
- `Testbarkeit der Anforderungen:` Lassen sich die Anforderungen in konkrete Testfälle überführen?

Nach Überprüfung wurden alle 11 Anforderungen analysiert. Während einige Anforderungen lediglich präzisiert wurden, waren bei anderen inhaltliche Anpassungen erforderlich, um Unklarheiten zu beseitigen und die Testbarkeit zu geährleisten.

Von den überprüften 11 Anforderungen:

- 5 konnten unverändert übernommen werden,
- 3 wurden konkretisiert,
- 3 mussten inhaltlich angepasst werden `(z. B. neue Fehlermeldungen, Validierungsregeln)`.

Die vollständige Analyse mit konkreten Verbesserungsvorschlägen ist in folgender Tabelle dokumentiert:

Tabelle 2.1: Identifizierte Probleme und Verbesserungsvorschläge

Nr	Anforderung	Probelm/ Unklarheit	Verbesserungsvorschlag
1	Gebäudestruktur (1..n Gebäude, Eingänge, Wohnungen, Zähler)	Keine klare Definition von „Eingang“. Ist ein Eingang ein Gebäudeteil oder eine logische Struktur?	Definition eines Eingangs hinzufügen (z. B. „Ein Eingang ist eine physische oder logische Einheit, die Zugang zu Wohnungen ermöglicht.“).
2	Verschiedene Zählertypen (Strom, Gas, Wasser)	Unklar, ob weitere Typen ergänzbar.	Klarstellung, ob die Liste erweiterbar ist und wie neue Zählertypen ergänzbar.
3	Zähler-ID	Keine Vorgabe zur Länge oder zum Format der ID.	Die Zähler-ID muss eindeutig sein und darf nicht mehrfach vergeben werden. Die ID wird automatisch nach dem Schema Gebäude-Jahr-Random generiert.
4	Datenfilterung	Unklar, welche Filtermöglichkeiten existieren (Gebäude, Zeitraum?).	Ergänzung von Filtern nach Gebäude, Wohnung, Zeitraum und Zählertyp.
5	Ablesewerte	Unklar, ob rückwirkende Korrekturen möglich sind.	Spezifikation: Ablesewerte können nur in der Zukunft oder am aktuellen Tag eingetragen werden. Änderungen nur durch Admins.
6	Zähler sind über ihre ID zu finden	Was passiert, wenn eine ID nicht existiert?	Falls eine Zähler-ID nicht existiert, erscheint die Fehlermeldung: ’Ungültige Zählernummer. Bitte überprüfen Sie Ihre Eingabe
7	Zähler sollen abgelesen werden (Eingabe von Datum und Wert)	Gibt es eine Validierung für vergangene/future Daten?	Klarstellung, ob das Ablesedatum nur in der Vergangenheit oder auch in der Zukunft liegen darf.
8	Zähler und Datum laufen nur vorwärts	Fehlt eine Angabe zu Testfällen (z. B. wie rückdatierte Werte behandelt werden)	Testfälle für Grenzwerte (min/max Werte für Datum) spezifizieren
9	Weitere Ablese-informationen eingeben (Ablesung, Schätzung)	Müssen Nutzer einen Ablesetyp zwingend angeben oder gibt es Standardwerte?	Standardwert oder Pflichtfeld definieren.
10	Ableser-Informationen eingeben (Hauswart, Mieter, Energie-versorger)	Können mehrere Ableser für einen Zähler existieren?	Klärung, ob Mehrfachzuweisungen erlaubt sind.
11	Verbrauch berechnen und anzeigen	Sind historische Verbrauchswerte abrufbar?	Die Verbrauchsanzeige wird nach jeder neuen Ablesung automatisch aktualisiert. Keine manuelle Aktualisierungist erforderlich. Historische Verbrauchsdaten werden für mindestens 12 Monate gespeichert.

Verantwortliche Personen und Datum

- Junior Lesage Ekane Njoh
- Franck Majesté Silatsa Dogmo
- Datum: 20.02.2025

2.2 Verbesserung der Anforderungen

Auf Basis unseres Reviews konnten wir die Anforderungen an das Hausverwaltungsprojekt verbessern. Dabei wuurdenn unklare Definitionen konkretisiert, Testbarkeit verbessert und Validierungsregeln ergänzt.

im Vergleich zu den ursprünglichen Anforderungen haben sich insbesondere die folgenden Aspekte geändert:

- Definition der Zähler-ID (eindeutig, 14-stellig, festes Format)
- Neue Fehlerbehandlungen für ungültige ID-Eingaben
- Validierungsregeln für vergangene und zukünftige Ablesewerte
- Optimierung der Verbrauchsanzeige mit Berücksichtigung fehlender Werte
- Skalierbarkeit für größere Datenmengen mit 5000+ Zählern

Nach Überlegung fannden wir es gut funktionalen von nicht funktionalen Anforderungen zu trennen. Die Trennung zwischen funktionalen und nicht-funktionalen Anforderungen ist essenziell, um eine klare Strukturierung der Systemanforderungen zu gewährleisten.

Unsere funktionale Anforderungen definieren was das System tun soll, also welche konkreten Funktionen es bereitstellt. Sie sind direkt testbar und beschreiben die Interaktionen zwischen Nutzern und System. Nicht-funktionale Anforderungen hingegen spezifizieren wie das System diese Funktionen bereitstellen soll, also Qualitätsmerkmale wie Performance, Benutzerfreundlichkeit oder Skalierbarkeit. Durch diese Trennung wird es einfacher, sowohl die funktionale Umsetzung als auch die technischen Rahmenbedingungen des Prototyps gezielt zu überprüfen und zu optimieren.

2.2.1 Funktionale Anforderungen

Die folgende Tabelle enthält die funktionalen Anforderungen unseres Hausverwaltungsprototyps. Diese Anforderungen legen fest, welche Funktionen das System bieten muss, um eine effektive Verwaltung von Gebäuden, Zählern und Verbrauchsdaten zu ermöglichen. Dazu gehören unter anderem das Erfassen von Zählerständen, die Filterung von Daten sowie die Berechnung und Anzeige des Verbrauchs. Jede Anforderung ist so formuliert, dass sie klar verständlich und testbar ist.

Tabelle 2.2: Funktionale Anforderungen

Nr.	Anforderung	Beschreibung
F1	Gebäudestruktur verwalten	Gebäude können mehrere Eingänge haben, jede Wohnung hat eine eindeutige ID.
F2	Zählertypen verwalten	Unterstützte Typen: Strom, Gas, Wasser. Die Liste ist erweiterbar, indem neue Typen über eine Konfigurationsdatei durch Entwickler hinzugefügt werden.
F3	Zählerverwaltung	Jeder Zähler hat eine eindeutige ID im Format `Gebäude-Jahr-Random` (14-stellig). Jeder Zähler gehört zu einer Wohnung und einem Zählertyp. Er speichert den letzten Ablesewert, das letzte Ablesedatum und die Ablesemethode.
F4	Datenfilterung	Filter nach Gebäude, Wohnung, Zählertyp und Zeitraum.
F5	Zählerablesung	Zählerwerte können nur mit aktuellem oder zukünftigen Datum erfasst werden. Negative Werte sind nicht zulässig. Falls der neue Wert kleiner als der vorherige ist, gibt es eine Fehlermeldung. Admins können jedoch rückwirkende Korrekturen vornehmen, falls ein Fehler festgestellt wird.
F6	Fehlermeldungen	Falls eine Zähler-ID nicht existiert, erscheint „Die eingegebene ID existiert nicht“. Falls eine Wohnung keiner ID zugeordnet ist, erscheint „Dieser Zähler ist keiner Wohnung zugeordnet.“
F7	Verbrauchsanzeige	Historische Verbrauchswerte sind für die letzten 12 Monate abrufbar. Eine grafische Darstellung ist möglich.
F8	Ableser-Informationen	Ableser können Hauswart, Mieter oder Energieversorger sein. Falls keine Information vorhanden ist, wird „Unbekannt“ eingetragen.
F9	Bearbeiten und Löschen von Gebäuden	Gebäude können direkt bearbeitet oder gelöscht werden.
F10	Zurück-Buttons auf allen Seiten	Verbesserte Navigation in der Anwendung.
F11	Gebäude auswählen vor Verbrauchsanzeige	Nutzer müssen erst ein Gebäude wählen, bevor Verbrauchsdaten angezeigt werden.
F12	Direkte Weiterleitung bei nur einem Gebäude	Wenn nur ein Gebäude existiert, wird die Verbrauchsanzeige sofort geladen.
F13	Unterschiedliche Speicherung für aktuelle	historische Verbrauchsdaten: verbrauch_aktuell_X.png und `verbrauch_historie_X_YYYY-MM-DD.png` werden getrennt gespeichert.

2.2.2 Nicht-funktionale Anforderungen

Neben der funktionalen Umsetzung muss das System bestimmte nicht-funktionale Anforderungen erfüllen. Diese betreffen Aspekte wie Systemperformance, Skalierbarkeit, Fehlerbehandlung und Benutzerfreundlichkeit. Während funktionale Anforderungen definieren, „was“ das System tun soll, beschreiben nicht-funktionale Anforderungen, „wie gut“ es das tun muss. Besonders wichtig sind hier Antwortzeiten der Verbrauchsanzeige, die visuelle Darstellung der Verbrauchsdaten sowie Datenschutzaspekte im Umgang mit Zählerwerten.

Tabelle 2.3: Nicht-Funktionale Anforderungen

Nr.	Anforderung	Beschreibung
NF1	Zeitraum für die Verbrauchsanzeige im Diagramm sichtbar	Das Diagramm zeigt den Zeitraum der Messung an (z. B. „März 2024 - Februar 2025“)und wird automatisch aktualisiert, sobald neue Verbrauchsdaten eingegeben werden.
NF2	Letzte 12 Monate immer anzeigen (auch ohne Werte)	Die Verbrauchsanzeige berücksichtigt automatisch die letzten 12 Monate. Fehlende Werte werden als „0“ dargestellt.
NF3	Farbliche Kennzeichnung der Zähler in der Verbrauchsanzeige	Jeder Zähler erhält eine eindeutige Farbe zur besseren Unterscheidung.
NF4	Optimierung der Antwortzeiten	Das System soll Verbrauchsdaten in unter 2 Sekunden berechnen und anzeigen. Die Berechnung muss auch bei einer Last von 5000 Zählern stabil bleiben.
NF5	Datenintegrität und Konsistenz	Ablesewerte dürfen nicht rückwirkend geändert werden (außer durch Admins).
NF6	Speicherung von Verbrauchsdaten gemäß Datenschutzbestimmungen	Verbrauchsdaten dürfen nur von autorisierten Nutzern eingesehen werden.
NF7	System skalierbar für große Datenmengen	Unterstützung für mindestens 100 Gebäude und 5000 Zähler.

3 Testkonzept

Ein strukturiertes Testkonzept ist essenziell, um die Qualität und Stabilität der entwickelten Hausverwaltungssoftware sicherzustellen. Da es sich um einen Prototype handelt, fokussieren wir uns auf technische Tests zur Funktionsprüfung und verzichten auf umfassende Usability- oder Systemtests. Unser Ziel ist es, sicherzustellen, dass die Kernfunktionen fehlerfrei arbeiten, Daten korrekt verarbeitet werden und das System auch unter Last stabil bleibt. Die Tests orientieren sich an etablierten Softwaretestverfahren und wurden so konzipiert, dass sie eine möglichst hohe Abdeckung der Anforderungen gewährleisten.

3.1 Auswahl von Testverfahren

Wir wissen aus der Vorlesung, dass sich Softwaretests grundsätzlich in drei Kategorien unterteilen lassen:

- **White-Box-Testing:** Interne Logik der Software wird geprüft, Code-Abdeckung ist entscheidend.
- **Black-Box-Testing:** Tests erfolgen ohne Kenntnis des Quellcodes, Fokus liegt auf den Ein- und Ausgaben des Systems.
- **Gray-Box-Testing:** Kombination aus White-Box und Black-Box, teilweise Kenntnisse über den Code werden verwendet.

Basierend auf den Anforderungen der Hasuverwaltung haben wir uns für eine Kombination aus **Black-Box-Testing** und **Gray-Box-Testing** entschieden. Besonders relevant waren für uns folgende Techniken:

- **Äquivalenzklassenbildung:** Gruppierung von Eingabewerten, um mit möglichst wenigen Tests eine hohe Abdeckung zu erzielen.
- **Boundary-Value-Testing:** Überprüfung von Grenzwerten (z. B. Mindest- und Höchstwerte für Ablesungen).

Dass wir die Techniken bei uns hervorgehoben wurde, lässt sich dadurch erklären, dass unser Prottyp eine Webanwendung ist, was dazu führt, dass der Fokus hier eher auf der Funktionalitätder Schnittstellen und Datenverarbeitung liegt. Außerdem reduziert die Äquivalenzklassenbildung die Anzahl der Testfälle, während trotzdem eine breite Abdeckung erreicht wird. Letzendlich ist Boundary-value-Testing essenziell für die Überprüfung von Verbrauchswerten und Ablesedaten.

3.2 Testziele und Strategie

Für unser Testkonzept haben wir uns klare Testziele definiert, damit wir den Fokus behalten können und nicht unnötig Zeit auf Vorgänge investieren, die für das Projekt irrelevant sind.

Wir haben folgende Ziele verfolgt:

- Sicherstellen, dass die Kernfunktionen der Software korrekt arbeiten.
- Überprüfung, ob Module und Komponenten korrekt zusammenarbeiten.
- Validierung der Fehlerbehandlung durch gezielte Eingabe ungültiger Werte.
- Sicherstellung der Performance unter hoher Last.

Damit die Testumsetzung für uns noch leichter wird, haben wir uns Teststrategien überlegt. Unser Testkonzept folgt einer `Botton-Up` Strategie, bei der zunächst einzelne Komponente geprüft und anschließend System- und Performancetests durchgeführt werden:

- **Unit-Tests**: Isolierte Tests einzelner Funktionen (z. B. Validierung von Zähler-IDs oder Verbrauchsdaten).
- **Funktionstests**: Prüfung der Geschäftslogik, u. a. Zählerverwaltung, Ablesungen und Verbrauchsberechnung.
- **Performance-Tests**: Simulation von hoher Last, um die Skalierbarkeit zu überprüfen.
- **Negative Tests**: Überprüfung der Fehlerbehandlung durch ungültige Eingaben.

3.3 Testumgebung und Testdaten

3.3.1 Testumgebung

Unsere Testumgebung haben wir versucht, so einfach wie möglich zu halten, damit wir nicht den Rahmen überspringen. Hier sind die Kernpunkte unserer Testumgebung aufgelistet:

- Der Prototyp wird in einer lokalen Entwicklungsumgebung als Flask-Anwendung entwickelt und getestet.
- Die Tests werden mithilfe von pytest automatisiert durchgeführt.
- Performance-Tests erfolgen durch Simulation hoher Anfragen über eine Flask-Testumgebung.
- Erstellung von Testfällen erfolgt nach den Prinzipien von Äquivalenzklassenbildung und Grenzwertanalyse.

3.3.2 Testdaten

Zur Testen gehören auch Testdaten zur Simulation, weil wir noch bei einem Prototyp sind, dessen Einsatz in einer produktiven Umgebung geplant ist.

- Eine Testdatenbank mit Dummy-Daten wird verwendet.
- Persistenz der Daten erfolgt über JSON-Dateien.
- Für Lasttests werden 1000 simultane Ablesungen simuliert.
- Testfälle für Grenzwerte und ungültige Werte (z. B. negative Ablesungen) wurden vorbereitet.

3.3.3 Ausgewähllte Testverfahren

Tabelle 3.1: Ausgewählte Testverfahren

Testverfahren	Einsatzbereich	Begründung
Unit-Tests	Einzelne Funktionen wie Datenvalidierung, ID-Format, Speicherung von Ablesewerten	Frühes Erkennen von Fehlern in einzelnen Modulen.
Funktionstests	Überprüfung der gesamten Funktionalität wie Zählerverwaltung, Ablesungen, Filterung, Verbrauchsanzeige	Sicherstellung der korrekten Umsetzung der Anforderungen.
Performance-Tests	Simulation hoher Last durch 1000+ gleichzeitige Ablesungen	Sicherstellung, dass das System auch mit vielen Gebäuden und Zählern performant bleibt.
Negative Tests	Eingabe ungültiger Werte (z. B. leere Felder, falsche ID, negatives Datum)	Überprüfung der Fehlerbehandlung und Robustheit des Systems.

Mit diesem Testkonzept soll erreicht werden, dass die Kernfunktionen des Prototyps validiert werden, ohne unnötig komplexe oder produktionstaugliche Tests durchzuführen. Die Kombination aus **Unit-Tests, Funktionstests, Performance-Tests und Negativen Tests** bietet eine solide Basis für die Qualitätssicherung. Mit diesem Ansatz wird überprüft, ob das System stabil und fehlerresistent ist, sowie unter hoher Last zuverlässig arbeitet. Die Testergebnisse werden in einer strukturierten Form dokumentiert, um Erkenntnisse für mögliche Optimierungen des Prototyps zu gewinnen.

4 Entwicklung der Testfälle

4.1 Ableitung konkreter Testfälle

4.2 Erstellung einer Testfall-Dokumentation

5 Prototypische Umsetzung der Hausverwaltung

5.1 Software-Architektur und Technologien

5.2 Implentierung

5.3 Anwendung des Testkonzepts

6 Qualitätsmanagement-Methoden in der Softwareentwicklung

6.1 Relevanz der Qualitätssicherung

6.2 Anwendung von QS-Methoden im Projekt

7 Fazit

Listingverzeichnis

Anhang

Selbstständigkeitserklärung

Wir versichern, die von uns vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die wir für die Arbeit benutzt haben, sind angegeben. Die Arbeit haben wir mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegt.

Bremerhaven, den 5. März 2025 Unterschrift:

Junior Leage EKane Njoh, Franck Majeste
Silatsa Dogmo, Steve Aguiwo II