

# SMART HEALTH JEWELRY

## Gruppenmitglieder

Dennis Apelt

Maxime Riebschläger

Eduard Wegele

Linus Brüstle



Projekt I  
Dozentin: Sofia Toto  
09.05.2024

# Inhaltsverzeichnis

Einleitung .....	2
Was ist eine Smart ID?.....	2
Wie kann man die Heilchancen/Überlebenschancen von älteren Menschen erhöhen?.....	3
Was beinhaltet diese Smart ID? .....	3
Stakeholder Analyse .....	4
Hauptteil .....	6
Projektmanagement Methoden .....	6
Designentwürfe der Brosche .....	9
Praktische Implementierung der Hardware .....	10
Funktionsweiser der Sensoren .....	11
Ablauf des Arduino Codes .....	13
Klassenstruktur der API für Sensordaten .....	15
Software Features .....	16
Kommunikation .....	25
E-Mail .....	26
Erweiterung der Brosche durch GUI-Applikation .....	28
1) Vorstellung.....	28
2) Design der GUI-Desktop Applikation .....	28
3) Erstellung / Programmierung des Designs .....	31
4) Testen einzelner Screens .....	34
Daten .....	40
Datenweiterverarbeitung.....	41
Datenschutzkonformität.....	42
Ethische Bedenken .....	42
Schluss .....	43
Rückblick auf das Projekt .....	43
Zukünftige Optimierungen.....	44
Eigenständigkeitserklärung .....	45

# Einleitung

In einer Gesellschaft, in der die Gesundheit älterer Menschen stets von hoher Priorität ist, eröffnen sich durch technologische Innovationen neue Wege, um die Gesundheitsversorgung zu verbessern und das Wohlbefinden zu fördern. Insbesondere ältere Menschen sind anfällig für verschiedene Krankheiten und Gesundheitsrisiken, die eine kontinuierliche Überwachung und rechtzeitige Intervention erfordern. In diesem Kontext stellt Smart Jewelry eine vielversprechende Lösung dar, um die Gesundheitsdaten älterer Menschen zu erfassen, zu analysieren und im Bedarfsfall lebensrettende Maßnahmen einzuleiten.

Diese Dokumentation widmet sich der Entwicklungs- und Funktionsweise einer innovativen Brosche, die nicht nur als Schmuckstück dient, sondern auch als intelligentes medizinisches Gerät fungiert. Durch die Integration von Hochtechnologie ermöglicht diese Brosche das präzise Messen verschiedener Vitalwerte und bietet eine Vielzahl von Funktionen, die darauf abzielen, die Gesundheit und Sicherheit älterer Menschen zu verbessern.

## Was ist eine Smart ID?

In der heutigen digitalen Welt spielen sichere und bequeme Identitätslösungen eine entscheidende Rolle für den Schutz persönlicher Daten und die Gewährleistung reibungsloser Online-Transaktionen. Smart ID, auch bekannt als digitale Identität oder elektronische Identität, ist ein innovatives Konzept, das darauf abzielt, diese Anforderungen zu erfüllen.

Im Kern ist eine Smart ID eine digitale Version Ihrer physischen Identität, die auf einem sicheren Datenträger wie einer Smartcard oder einem Smartphone gespeichert wird. Sie enthält persönliche Informationen über den Nutzer, wie den Namen, Geburtsdatum und Adresse. Eine Smart ID kann verwendet werden, um sich online zu authentifizieren, auf Dienste zuzugreifen und Transaktionen abzuschließen.

Zu den Vorteilen der Smart ID gehören eine erhöhte Sicherheit, welche durch Verschlüsselungstechnologien und Authentifizierungsverfahren, die das Risiko von Identitätsdiebstahl und Betrug erheblich verringern, hergestellt werden. Ein weiterer Vorteil ist die Verbesserte Benutzerfreundlichkeit. Smart IDs ermöglichen es Benutzern, sich bequem und einfach online zu authentifizieren, ohne Passwörter eingeben oder sich an Anmeldedaten erinnern zu müssen.

Smart IDs können außerdem für die sichere Abwicklung von Online-Transaktionen wie Einkäufen, Bankgeschäften und E-Government-Diensten verwendet werden.

## Wie kann man die Heilchancen/Überlebenschancen von älteren Menschen erhöhen?

Die Implementierung einer Vielzahl von Sensoren in der Brosche für ältere Menschen eröffnet neue Möglichkeiten, ihre Gesundheit proaktiv zu überwachen und im Notfall schnell Hilfe zu erhalten. Die Smart Jewelry beinhaltet dabei folgende Lösungen:

### 1. Frühzeitige Notfallerkennung und -reaktion:

Die Brosche ist mit Sensoren ausgestattet, die Stürze erkennen und automatisch einen Notruf absetzen können. Dadurch wird im Falle eines Unfalls oder medizinischen Notfalls schnell Hilfe alarmiert, was die Chancen auf eine rechtzeitige medizinische Versorgung und damit das Überleben erhöht.

### 2. Kontinuierliche Gesundheitsüberwachung:

Die Brosche überwacht kontinuierlich Vitalparameter wie Blutdruck, Herzschlag. Dadurch können potenzielle gesundheitliche Probleme frühzeitig erkannt und behandelt werden, was die Heilchancen verbessert.

### 3. Medikamenten- und Terminmanagement:

Die Brosche erinnert ältere Menschen an die Einnahme von Medikamenten und wichtige Termine, was dazu beiträgt Komplikationen zu vermeiden.

### 4. Lokalisierung im Notfall:

Durch den integrierten GPS-Sensor kann die Position des Trägers genau verfolgt werden, was im Notfall die Lokalisierung und Rettung erleichtert und somit die Überlebenschancen erhöht.

### 5. Gesundheitsförderung durch Aktivitätsverfolgung und Schlafüberwachung:

Die Brosche verfolgt die Aktivitäten und den Schlaf des Trägers, was dazu beiträgt, einen gesunden Lebensstil zu fördern und das Risiko von Komplikationen zu verringern.

## Was beinhaltet diese Smart ID?

Die vorgestellte Smart ID besteht aus zwei Grundkomponenten. Die Hardwarekomponente besteht aus der frei individualisierbaren Brosche, welche mit mehreren Sensoren ausgestattet ist. Jeder Sensor erfüllt seine eigene spezialisierte Aufgabe. Die jeweiligen Teilbereiche werden im Hauptteil unter [Hardware Komponenten](#) detailliert beschrieben.

Die zweite Komponente spielt sich im Software Backend ab. Dort werden alle, von den Sensoren gesammelten Daten, gesammelt und analysiert. Um die Daten anschaulich zu gestalten, wurde eine eigene Handyapplikation erstellt, welche frei zugänglich über die jeweiligen Appstores sind. Der jeweilige Nutzer kann seine Daten selbst einsehen oder anderen Personen, die Möglichkeit geben, diese einzusehen. Dies ist sehr hilfreich, falls ein Pfleger

oder ein Familienangehöriger sich um den Nutzer kümmert. Im Falle eines außergewöhnlichen Wertes, wird eine Push-Benachrichtigung auf das Smartphone gesendet, um alle Personen zu benachrichtigen. Diese können dann auf die jeweilige Situation passend reagieren.

## Stakeholder Analyse

In dieser Analyse werden die Stakeholder im Kontext der Einführung von Smart Jewelry für ältere Menschen untersucht. Stakeholder stellen dabei alle Personen oder Gruppen dar, die ein Interesse an oder einen Einfluss auf das Projekt haben könnten.

### Ältere Menschen und ihre Familien:

Ältere Menschen sind die primären Nutzer der Smart Jewelry. Sie haben ein starkes Interesse an einer verbesserten Gesundheitsüberwachung und -versorgung, da dies ihre Lebensqualität und Sicherheit verbessern kann. Ihre Familienangehörigen sind ebenfalls Stakeholder, da sie sich um das Wohlergehen der älteren Menschen kümmern und von der Technologie profitieren können, um ihnen Unterstützung zu bieten.

### Gesundheitsdienstleister:

Ärzte, Krankenschwestern und andere Gesundheitsdienstleister sind wichtige Stakeholder, da sie von den Daten profitieren können, die von der Smart Jewelry gesammelt werden. Sie haben ein Interesse daran, dass ihre Patienten besser überwacht werden und rechtzeitig medizinische Interventionen erhalten, was die Qualität der Gesundheitsversorgung verbessert.

### Technologieunternehmen und Entwickler:

Unternehmen, die die Smart Jewelry entwickeln, herstellen und warten, sind entscheidende Stakeholder. Sie haben ein finanzielles Interesse am Erfolg des Produkts und sind verantwortlich für die Bereitstellung von qualitativ hochwertigen und zuverlässigen Technologien, die den Bedürfnissen der Nutzer entsprechen.

### Regulierungsbehörden und Gesetzgeber:

Regierungsbehörden und Gesetzgeber sind Stakeholder, da sie für die Regulierung und Zulassung medizinischer Geräte und persönlicher Gesundheitsdaten verantwortlich sind. Sie haben ein Interesse daran, sicherzustellen, dass die Smart Jewelry den geltenden Standards und Vorschriften entspricht, um die Sicherheit und Privatsphäre der Nutzer zu gewährleisten.

### Versicherungsunternehmen:

Versicherungsunternehmen könnten Stakeholder sein, da sie ein Interesse an der Nutzung von Technologien haben, die zur Prävention von Gesundheitsproblemen beitragen und die Kosten für medizinische Behandlungen senken können. Sie könnten Anreize für die Nutzung von Smart Jewelry durch Prämienrabatte oder andere Vorteile bieten.

### Interessengruppen und NGOs:

Organisationen, die sich für die Rechte und das Wohlergehen älterer Menschen einsetzen, könnten Stakeholder sein. Sie könnten ein Interesse daran haben, sicherzustellen, dass die Smart Jewelry die Bedürfnisse und Rechte älterer Menschen respektiert und dass sie für alle Bevölkerungsgruppen zugänglich ist.

Insgesamt zeigt die Stakeholder-Analyse, dass die erfolgreiche Einführung von Smart Jewelry für ältere Menschen eine umfassende Berücksichtigung der Bedürfnisse und Erwartungen verschiedener Interessengruppen erfordert. Indem wir die Perspektiven der Stakeholder verstehen und ihre Anliegen ansprechen, können wir sicherstellen, dass das Projekt nicht nur technologisch erfolgreich ist, sondern auch einen positiven Einfluss auf die Gesundheitsversorgung älterer Menschen hat und ihre Lebensqualität verbessert.

# Hauptteil

## Projektmanagement Methoden

Für das effektive Management eines Projekts sind verschiedene Methoden und Werkzeuge entscheidend. Zu den angewandten Methoden in unserem Projekt zählen der GANTT-Chart, Trello und der Projektauftrag. Jede dieser Methoden spielt eine wichtige Rolle bei der Planung, Organisation und Durchführung unserer Projekte. Der GANTT-Chart ermöglicht eine visuelle Zeitplanung und Ressourcenverwaltung, während Trello als Scrum Sprint-Meetingsplanung und zur Aufgabenverwaltung dient. Der Projektauftrag legt die Ziele, den Umfang und die Rahmenbedingungen des Projekts fest und dient als grundlegender Leitfaden für alle Projektbeteiligten. Durch die gezielte Anwendung dieser Methoden konnten wir unser Projekt erfolgreich und reibungslos durchführen.

### Projektauftrag: Smart Health Jewelry

Projektname:	Smart Health Jewelry		
Projektorganisation			
Auftraggeber	Sofia Toto, Lehrbeauftragte Hochschule Offenburg		
Projektmitglieder	Dennis Apelt, Linus Brüstle, Maxime Riebschläger, Eduard Wegele		
Projektdauer	10 Wochen		
Start- und Endtermin	01.05.2024 – 10.07.2024		
Termine	10.07.2024 Abgabe Dokumentation Ende Juli 2024 Ergebnispräsentation		
Meilensteine	1. personalisierbare Designvorschläge der Brosche erstellen 2. Funktionalitäten der Brosche definieren 3. Auswahl der technischen Komponenten (Anschaffung), Architekturplanung von Hard- und Software 4. Implementierung der technischen Komponenten mit Schnittstelle zur Applikation 5. Datenbeschaffung und Aufbereitung 6. Visualisierung der Daten auf einer Applikation 7. Daten auswerten anhand von Algorithmen, Benachrichtigung und Anzeige des Zustands 8. Dokumentation	08.05.2024 15.05.2024 22.05.2024  29.05.2024 12.06.2024 19.06.2024  26.06.2024 03.07.2024	
Projektauftrag			
Ausgangslage	Ältere Menschen sind anfällig für diverse Krankheiten. Ein Smart Jewelry ermöglicht hierbei das Messen der Vitalwerte und im Notfall verschiedene wichtige Schritte einzuleiten.		
Projektziele:	Design und Konzeptvorschläge in Hard- und Software unter Verwendung von SMART ID (Emoji-Anzeige auf einer Brosche). Hardware beinhaltet Wahl der relevanten Sensoren. Software sorgt dabei für ein reibungsloses Zusammenspiel aller Beteiligten an der Ereigniskette nach einem Notfall. Ziel ist es Angehörige, Ärzte und Beteiligte zeitnah über das Auftreten von Komplikationen zu benachrichtigen, sodass schnell gehandelt werden kann.		
Projektumfang	Entwurf-, Design und Architekturvorschläge einer Brosche, Sensorsauswahl, Schnittstellen zwischen Hard- und Software, (Pseudo-) Datenaufbereitung und Analyse, Anzeige Gesundheitszustand, Benachrichtigungsmitteilung		
Abgrenzungen:	-		
Problembereiche	-		
Projektergebnis	Ausgearbeitete Design- und Architekturvorschläge für die Hardware sowie Applikation zur Simulation einer Ereigniskette		

Abbildung 1 – Projektauftrag

Aufgabe	Startdatum	Enddatum	Status	Teammitglieder	KW - 17	KW - 18	KW - 19	KW - 20	KW - 21	KW - 22	KW - 23	KW - 24	KW - 25	KW - 26	KW - 27	KW - 28	
<b>Vorarbeit</b>																	
personalisierbare Designvorschläge der Brosche erstellen	26.04.2024	03.05.2024	fertig	alle													
Funktionalitäten der Brosche definieren	26.04.2024	03.05.2024	fertig	alle													
Auswahl der technischen Komponenten (Anschaffung), Architekturplanung von Hard- und Software	26.04.2024	03.05.2024	fertig	Dennis													
Erstellung eines Softwarekonzeptes für das Backend mithilfe eines Klassendiagramm	27.05.2024	23.06.2024	fertig	Eduard													
<b>Implementierung</b>																	
Arduino Hardware Sensoren anschließen	17.05.2024	17.05.2024	fertig	Dennis													
Arduino Analogsignale Sensoren umwandeln	27.05.2024	27.05.2024	fertig	Dennis													
Arduino Verbindung mit API über Schnittstelle einrichten	27.05.2024	01.06.2024	fertig	Maxime, Dennis													
OO Implementierung der Klassen für Messwerte in API	31.05.2024	01.07.2024	fertig	Maxime, Dennis													
Auswertung der Daten in API	01.06.2024	03.07.2024	fertig	Maxime, Dennis													
Visualisierung der Daten auf einer Applikation	01.06.2024	04.07.2024	fertig	Maxime, Dennis													
Benachrichtigung und Anzeige des Zustands	16.06.2024	04.07.2024	fertig	Maxime													
Helligkeit der GUI an Umgebung anpassen	16.06.2024	16.06.2024	fertig	Maxime													
Implementierung Anmeldemaske	16.06.2024	04.07.2024	fertig	Maxime													
Implementierung der Klasseninfrastruktur des Backends	15.06.2024	04.07.2024	fertig	Eduard													
E-Mail/SMS Benachrichtigungssystem	23.06.2024	04.07.2024	fertig	Eduard													
<b>GUI</b>																	
Design aller GUI-Fenster in Figma	05.05.2024	08.07.2024	fertig	Linus													
Übertragen des Designs aus Figma in Microsoft Visual Studio Code	03.06.2024	09.07.2024	fertig	Linus													
<b>Dokumentation</b>																	
1. Einleitung	03.05.2024	09.07.2024	fertig	alle													
1.1 Problem, welches das Projekt behandelt	07.05.2024	07.05.2024	fertig	Maxime, Dennis													
1.2 Was ist eine Smart ID?	07.05.2024	07.05.2024	fertig	Maxime													
1.3 Wie kann man die Heilchancen/Überlebenschancen von älteren Menschen erhöhen?	07.05.2024	07.05.2024	fertig	Dennis													
1.4 Was beinhaltet diese Smart ID?	07.05.2024	07.05.2024	fertig	Maxime													
1.5 Stakeholder Analyse	07.05.2024	07.05.2024	fertig	Eduard													
2. Hauptteil																	
2.1 Projektmanagement Methoden	07.07.2024	07.07.2024	fertig	Dennis													
2.2 Designentwürfe der Brosche	07.05.2024	07.05.2024	fertig	Dennis													
2.3 Praktische Implementierung der Hardware	07.05.2024	01.07.2024	fertig	Dennis													
2.3.1 Anschluss und verwendete Sensoren in Arduino	02.06.2024	02.06.2024	fertig	Dennis													
2.3.2 Funktionsweise der Sensoren	22.06.2024	22.06.2024	fertig	Dennis													
2.3.3 Ablauf des Arduino Codes	24.06.2024	24.06.2024	fertig	Dennis													
2.3.4 Klassenstruktur der API für Sensordaten	25.06.2024	30.06.2024	fertig	Eduard, Dennis													
2.3.5 API Softwarefeatures	25.06.2024	05.07.2024	fertig	Maxime, Dennis													
2.3.6 Kommunikation	22.06.2024	08.07.2024	fertig	Eduard													
2.3.7 Email	22.06.2024	08.07.2024	fertig	Eduard													
2.4 GUI	28.06.2024	28.06.2024	fertig	Linus													
2.5 Datenverarbeitung	15.06.2024	30.06.2024	fertig	Eduard													
2.6 Ethische Bedenken	22.06.2024	08.07.2024	fertig	Linus													
3. Schluss																	
3.1 Fazit	01.07.2024	08.07.2024	fertig	Eduard													
3.2 Eigenständigkeitserklärung	01.07.2024	08.07.2024	fertig	Eduard													

Abbildung 2 - GAANT-Chart

## GAANT Chart:

Ein Gantt-Diagramm ist ein unverzichtbares Werkzeug in unserem Projektmanagement, das es uns ermöglicht, den Projektzeitplan und den Fortschritt in Echtzeit zu verfolgen. Unser Gantt-Diagramm ist in die Phasen Vorarbeit, Implementierung, GUI und Dokumentation unterteilt. Jede Aufgabe innerhalb dieser Phasen ist mit Start- und Enddatum sowie dem aktuellen Status ("in Bearbeitung" oder "fertig") versehen, sodass Teammitglieder den Fortschritt live sehen können. Aufgaben werden den jeweiligen Teammitgliedern zugeordnet und die Kalenderwochen (KW) sind markiert, um den Aufwand visuell darzustellen. So können alle Beteiligten den Überblick behalten und die Effizienz des Projekts sicherstellen.

## Trello Board Chart:

Im Rahmen unseres Projekts haben wir Trello als zentrales Werkzeug für das Projektmanagement eingesetzt, insbesondere um agile Methoden wie Scrum zu unterstützen. Trello erwies sich als äußerst nützlich für die Organisation unserer Arbeitsabläufe, die Ideensammlung und die Aufgabenverteilung. Wir begannen mit der Nutzung von Trello, um regelmäßige Meetings gemäß dem Scrum-Ansatz zu planen und durchzuführen. Diese Meetings umfassten Sprint-Planungen, in denen wir mithilfe von Trello die Features auswählten, die während des nächsten Sprints umgesetzt werden sollten. Durch die einfache Verwaltung des Feature-Backlogs konnten wir Prioritäten setzen und Aufgaben effizient verteilen. Darüber hinaus nutzten wir Trello für unsere regelmäßige Meetings, bei denen wir den Fortschritt besprachen und etwaige Hindernisse identifizierten. Die Ideensammlung erfolgte ebenfalls über Trello, indem Teammitglieder Vorschläge einreichen und diskutieren konnten. Diese Funktion förderte die Kreativität und half uns, neue Features oder Verbesserungen zu identifizieren, die dann in das Projekt integriert wurden.

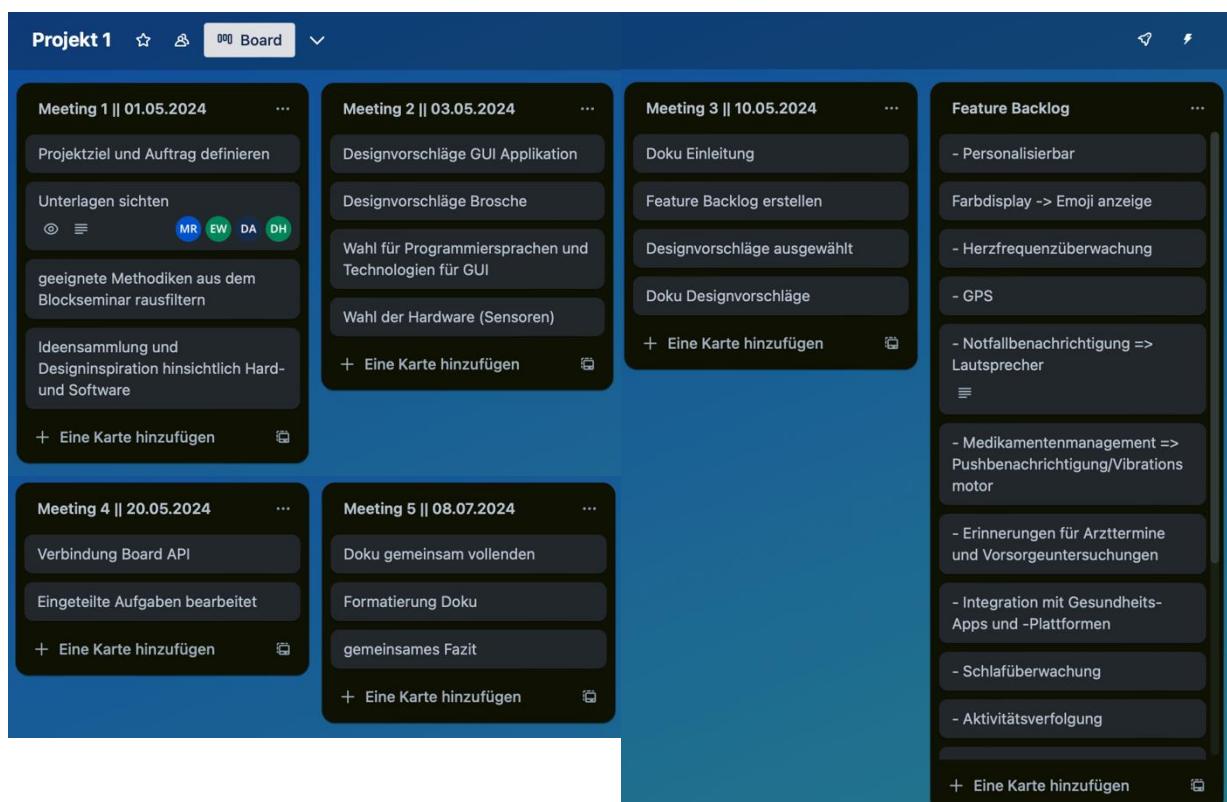


Abbildung 3 - Trello Board

## Designentwürfe der Brosche

Die Gestaltung der Brosche wurde von einem Ansatz inspiriert, der dem Benutzer die Möglichkeit bietet, sein persönliches Stilempfinden auszudrücken und sich flexibel an verschiedene Situationen anzupassen. Ein wesentliches Merkmal dieses Designs ist die Idee der Personalisierung und Individualisierung, die durch ein austauschbares Designelement realisiert wird. Dieses Element ermöglicht es dem Benutzer, das Aussehen der Brosche je nach Stimmung, Anlass oder Outfit anzupassen, was einen vielseitigen und ansprechenden ästhetischen Ausdruck gewährleistet.

Das Konzept der Personalisierung geht über bloße Ästhetik hinaus und bietet dem Benutzer ein Gefühl von Einzigartigkeit und Selbstausdruck. Durch die Möglichkeit, das Designelement der Brosche auszutauschen, wird ein Raum für Kreativität und Individualität geschaffen, der es dem Benutzer ermöglicht, seine Persönlichkeit und Vorlieben auf subtile, aber bedeutungsvolle Weise zum Ausdruck zu bringen.

Die Designinspiration hinter dieser personalisierbaren Brosche liegt in der Überzeugung, dass Schönheit nicht nur in der Äußerlichkeit liegt, sondern auch in der Möglichkeit, sich authentisch und ausdrucksstark zu präsentieren. Indem das Designelement der Brosche austauschbar gestaltet wurde, wird eine Plattform geschaffen, die es dem Benutzer ermöglicht, seiner Kreativität freien Lauf zu lassen und seine individuelle Identität zu betonen, während er gleichzeitig von der Funktionalität und Eleganz des Schmuckstücks profitiert.

### Designstile:

Das verzierte Erbstückdesign unserer Brosche vereint traditionelle Eleganz mit zeitloser Schönheit. Inspiriert von historischen Elementen und kunstvoller Handwerkskunst, ist jede Brosche ein einzigartiges Meisterwerk, das die Geschichte und Verbundenheit einer Familie verkörpert:

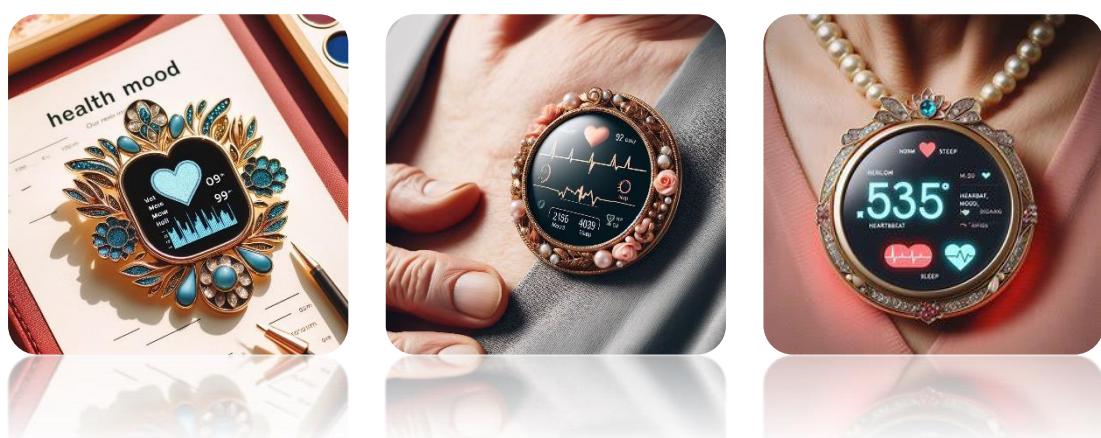


Abbildung 4 - Traditioneller Designstil

Unser modernes, schlichtes und zeitloses Designkonzept der Brosche vereint minimalistische Eleganz mit funktionaler Raffinesse. Klare Linien, reduzierte Details und hochwertige Materialien verleihen der Brosche eine zeitgemäße Ästhetik, die sich nahtlos in jede Lebenssituation einfügt:



Abbildung 5 - Moderner Designstil

## Praktische Implementierung der Hardware

Die Implementierung umfasst die Erfassung und Verarbeitung von Daten eines Herzschlagsensors, eines Temperatur- und Feuchtigkeitssensors, eines Thermistors, eines Fotowiderstands und eines Bewegungssensors.

### Verwendete Komponenten und Bibliotheken:

- Arduino Board (Arduino Uno, ESP32)
- GPIO Extension Board
- Breadboard
- DHT11 Hygrothermograph (Temperatur- und Feuchtigkeitssensor)
- NTC-Thermistor: Widerstandssensor zur Temperaturmessung
- Fotowiderstand: Lichtempfindlicher Widerstand
- MPU6050: Beschleunigungs- und Gyroskopsensor
- KY-039: Sensor zur Messung der Herzfrequenz
- 15 x Jumper Kabel
- 3 x 10KΩ Widerstand
- Arduino Bibliotheken:
  - SPI.h
  - Wire.h
  - DHTesp.h
  - MPU6050\_tockn.h

### Angeschlossene PINs:

- PIN\_SIGNAL\_HEARTBEAT 34

- PIN\_TEMP\_HUMID 32
- PIN\_THERMISTOR 2
- PIN\_PHOTORESISTOR 15
- PIN\_MOTION\_SDA 4
- PIN\_MOTION\_SCL 0

## Funktionsweiser der Sensoren

### **Herzschlagsensor (KY-039):**

Der Fototransistor in diesem Sensormodul arbeitet nach einem einfachen, aber effektiven Prinzip: Er funktioniert ähnlich wie ein gewöhnlicher Transistor, jedoch dient hier das einfallende Licht als Steuerspannung. Je mehr Licht auf den Fototransistor trifft, desto höher ist der durchgelassene Strom. Um dieses Verhalten zu messen, wird ein Widerstand in Reihe vor den Transistor geschaltet. So ergibt sich bei viel einfallendem Licht eine niedrige Spannung nahe 0V, während im Dunkeln eine Spannung nahe der Versorgungsspannung gemessen wird.

Im Kontext des Puls-Messmoduls mit Infrarotdiode und Fototransistor nutzt man dieses Prinzip zur Erkennung des Herzschlags. Wird ein Finger zwischen die Infrarotdiode und den Fototransistor gehalten, durchdringt das Infrarotlicht die Haut und erreicht den Fototransistor auf der anderen Seite. Die Haut und das darunter liegende Gewebe streuen das Licht, aber das Blut absorbiert es teilweise. Wenn das Herz Blut durch die Adern pumpt, verändert sich die Dichte des Blutes in den Gefäßen, was zu kleinen, aber messbaren Helligkeitsschwankungen führt.

Diese Helligkeitsschwankungen, die durch die wechselnde Lichtabsorption des pulsierenden Blutes entstehen, ändern den Strom durch den Fototransistor. Der Widerstand vor dem Fototransistor wandelt diese Stromänderungen in Spannungsschwankungen um, die am Signalausgang des Moduls detektiert werden können. So kann der Puls des Benutzers präzise gemessen werden.

### **Fotowiderstand:**

Ein Fotowiderstand ist ein lichtempfindlicher Widerstand, dessen Widerstand sich in Abhängigkeit von der auf seine lichtempfindliche Oberfläche treffenden Helligkeit verändert. Diese Änderung des Widerstands ermöglicht es, die Lichtintensität zu messen.

Wenn Licht auf den Fotowiderstand fällt, absorbiert das Material die Photonen und setzt Elektronen frei, was zu einer Verringerung des Widerstands führt. Bei Dunkelheit hat der Fotowiderstand einen hohen Widerstand, da weniger freie Elektronen zur Verfügung stehen. Diese charakteristische Veränderung des Widerstands in Abhängigkeit von der Lichtintensität ist das zentrale Funktionsprinzip des Fotowiderstands.

Um diese Veränderung zu messen, wird der Fotowiderstand in einer Schaltung typischerweise in Reihe mit einem festen Widerstand geschaltet, um einen Spannungsteiler zu bilden. Wenn

sich die Lichtintensität ändert, ändert sich der Widerstand des Fotowiderstands und damit die Spannung am Verbindungspunkt zwischen dem Fotowiderstand und dem festen Widerstand. Diese Spannung kann dann gemessen werden, um die Lichtintensität zu bestimmen.

### **NTC-Thermistor:**

Ein NTC-Thermistor (Negative Temperature Coefficient) ist ein temperaturabhängiger Widerstand, dessen Widerstand mit steigender Temperatur sinkt. Diese Funktionsweise beruht auf den Eigenschaften der verwendeten Metalloxide, die bei Temperaturänderungen ihre elektrische Leitfähigkeit verändern. Mit zunehmender Temperatur setzt die thermische Energie zusätzliche Ladungsträger frei, was den Widerstand verringert und den Stromfluss erleichtert.

Um die Temperaturänderung zu messen, wird der NTC-Thermistor in einer Spannungsteiler-Schaltung verwendet. Dabei ändert sich die Spannung am Verbindungspunkt zwischen dem Thermistor und einem festen Widerstand, wenn sich der Widerstand des Thermistors aufgrund einer Temperaturänderung ändert. Diese Spannung kann gemessen und zur Bestimmung der Temperatur genutzt werden.

### **Hygrothermograph (DHT11):**

Ein Hygrothermograph, wie der DHT11-Sensor, ist ein kombinierter Sensor, der sowohl Temperatur als auch Luftfeuchtigkeit misst. Der DHT11 nutzt hierfür zwei Hauptkomponenten: einen kapazitiven Feuchtigkeitssensor und einen Thermistor zur Temperaturmessung.

Der kapazitive Feuchtigkeitssensor besteht aus zwei Elektroden mit einem feuchtigkeitsempfindlichen Dielektrikum dazwischen. Bei Änderungen der Luftfeuchtigkeit ändert sich die Kapazität des Dielektrikums, was in ein elektrisches Signal umgewandelt wird. Der Thermistor im DHT11 ist ein temperaturabhängiger Widerstand, der seinen Widerstand mit der Temperatur ändert. Diese Widerstandsänderung wird in ein digitales Signal umgewandelt.

### **Beschleunigungs- und Gyroskopsensor (MPU6050)**

Der MPU6050 ist ein kombinierter Beschleunigungs- und Gyroskopsensor, der Bewegungsdaten in Echtzeit liefert. Er integriert einen 3-Achsen-Beschleunigungssensor und einen 3-Achsen-Gyroskopsensor in einem kompakten Bauteil.

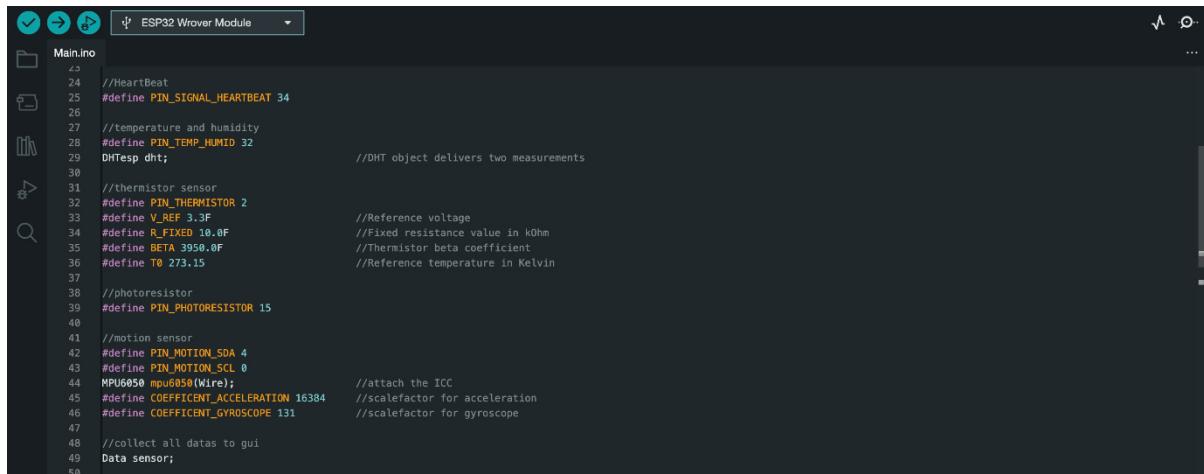
Der Beschleunigungssensor im MPU6050 misst die Änderungen der Geschwindigkeit entlang der X-, Y- und Z-Achsen. Dabei nutzt er mikro-elektro-mechanische Systeme (MEMS), die auf die mechanische Bewegung reagieren und diese in elektrische Signale umwandeln. Diese Signale werden digitalisiert und geben die Beschleunigung in Einheiten von g (Erdbeschleunigung) wieder.

Der Gyroskopsensor erfasst die Drehgeschwindigkeit oder Drehrate um die gleichen Achsen. Dies geschieht durch die Ausnutzung der Coriolis-Kraft, die auf die Sensorelemente wirkt,

wenn sie sich relativ zur Rotation bewegen. Die Ausgangssignale des Gyroskops werden ebenfalls digitalisiert und bieten die Messungen der Drehgeschwindigkeit in Grad pro Sekunde.

### Organisation im Arduino Code:

In der Programmierung ist die Verwendung symbolischer Konstanten eine bewährte Methode, um feste Werte durch benannte Bezeichner zu ersetzen. Dies verbessert die Lesbarkeit, Wartbarkeit und Fehlersicherheit des Codes erheblich. Im bereitgestellten Arduino-Code werden symbolische Konstanten, sowie Strukturen wie folgt definiert und verwendet:



```
ESP32 Wrover Module
Main.ino
24 //HeartBeat
25 #define PIN_SIGNAL_HEARTBEAT 34
26
27 //temperature and humidity
28 #define PIN_TEMP_HUMID 32
29 DHTesp dht; //DHT object delivers two measurements
30
31 //thermistor sensor
32 #define PIN_THERMISTOR 2
33 #define V_REF 3.3F //Reference voltage
34 #define R_FIXED 10.0F //Fixed resistance value in kOhm
35 #define BETA 3950.0F //Thermistor beta coefficient
36 #define T0 273.15 //Reference temperature in Kelvin
37
38 //photoresistor
39 #define PIN_PHOTORESISTOR 15
40
41 //motion sensor
42 #define PIN_MOTION_SDA 4
43 #define PIN_MOTION_SCL 0
44 MPU6050 mpu6050(Wire); //attach the IIC
45 #define COEFFICIENT_ACCELERATION 16384 //scaleFactor for acceleration
46 #define COEFFICIENT_GYROSCOPE 131 //scaleFactor for gyroscope
47
48 //collect all datas to gui
49 Data sensor;
```

Abbildung 6 - Organisation Arduino Code

## Ablauf des Arduino Codes

Die setup- und loop-Funktionen sind zentrale Bestandteile eines jeden Arduino-Programms (Sketch). Sie definieren, was einmalig beim Start des Programms und was wiederholt während der Laufzeit ausgeführt wird.

Setup erledigt folgende Hauptaufgaben:

- Start der seriellen Kommunikation:  
Ermöglicht die Ausgabe von Debug- und Sensordaten an den seriellen Monitor.
- Initialisierung der Sensoren:  
Konfiguriert die an das Arduino-Board angeschlossenen Sensoren
- Festlegung der Pin-Modi:  
Bestimmt, welche Pins als Eingänge oder Ausgänge fungieren.
- Start der I2C-Kommunikation:  
Initialisiert die Kommunikation für I2C-basierte Sensoren wie den MPU6050.

In der loop-Funktion werden alle Funktionen aufgerufen, die für die Abfrage und Verarbeitung der Daten von den verschiedenen Sensoren zuständig sind. Dabei wird eine Verzögerung von 10 Millisekunden zwischen den Aufrufen eingefügt, um sicherzustellen, dass die Schleife nicht zu schnell durchläuft. Dies hilft, die Sensordaten in einem angemessenen Intervall abzufragen und die Leistung des Mikrocontrollers nicht zu überlasten.

```

ESP32 Wrover Module

Main.ino

52
53 void setup()
54 {
55     Serial.begin(115200);
56     dht.setup(PIN_TEMP_HUMID, DHTesp::DHT11);
57     pinMode(PIN_SIGNAL_HEARTBEAT, INPUT);
58     //motion setup
59     Wire.begin(PIN_MOTION_SDA, PIN_MOTION_SCL);
60     mpu6050.begin();
61     mpu6050.calcGyroOffsets(true);
62 }
63
64 void loop()
65 {
66     delay(10);
67     vHeartBeatSensor();
68     vTemperatur_Humidity();
69     vThermistor();
70     vPhotoresistor();
71     vMotion();
72 }

```

Abbildung 7 - Setup- und Loop-Funktion

Die Daten der verschiedenen Sensoren werden in einem struct namens "Data" gespeichert. Diese Struktur ermöglicht es, alle relevanten Daten in einer einzigen Datenstruktur zu organisieren, was die Verwaltung und Übertragung erleichtert.

Die gesammelten Daten werden über die serielle Schnittstelle mit dem Komma (",") als Trennzeichen übertragen. Diese Art der Datenübertragung ermöglicht es, die Daten einfach zu parsen und zu interpretieren, sowohl von anderen Geräten als auch von einem Computer aus. Auf der Empfängerseite kann die empfangene Zeile anhand des Kommas in einzelne Datenfelder aufgeteilt werden, um sie weiter zu verarbeiten. Dieser Ansatz ermöglicht eine effiziente und gut lesbare Übertragung von Daten über die serielle Schnittstelle.

```

ESP32 Wrover Module

Main.ino

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <SPI.h>
5 #include "DHTesp.h"
6 #include <MPU6050_tockn.h>
7 #include <Wire.h>
8
9 struct Data
10 {
11     unsigned int usHeartBeat;
12     float fTemperature;
13     float fHumidity;
14     float fThermistor;
15     int iPhotoresistor;
16     float fx;
17     float fy;
18     float fz;
19     float fGyroX;
20     float fGyroY;
21     float fGyroZ;
22 };

```

Abbildung 8 - Organisation der Daten in Struktur

## Klassenstruktur der API für Sensordaten

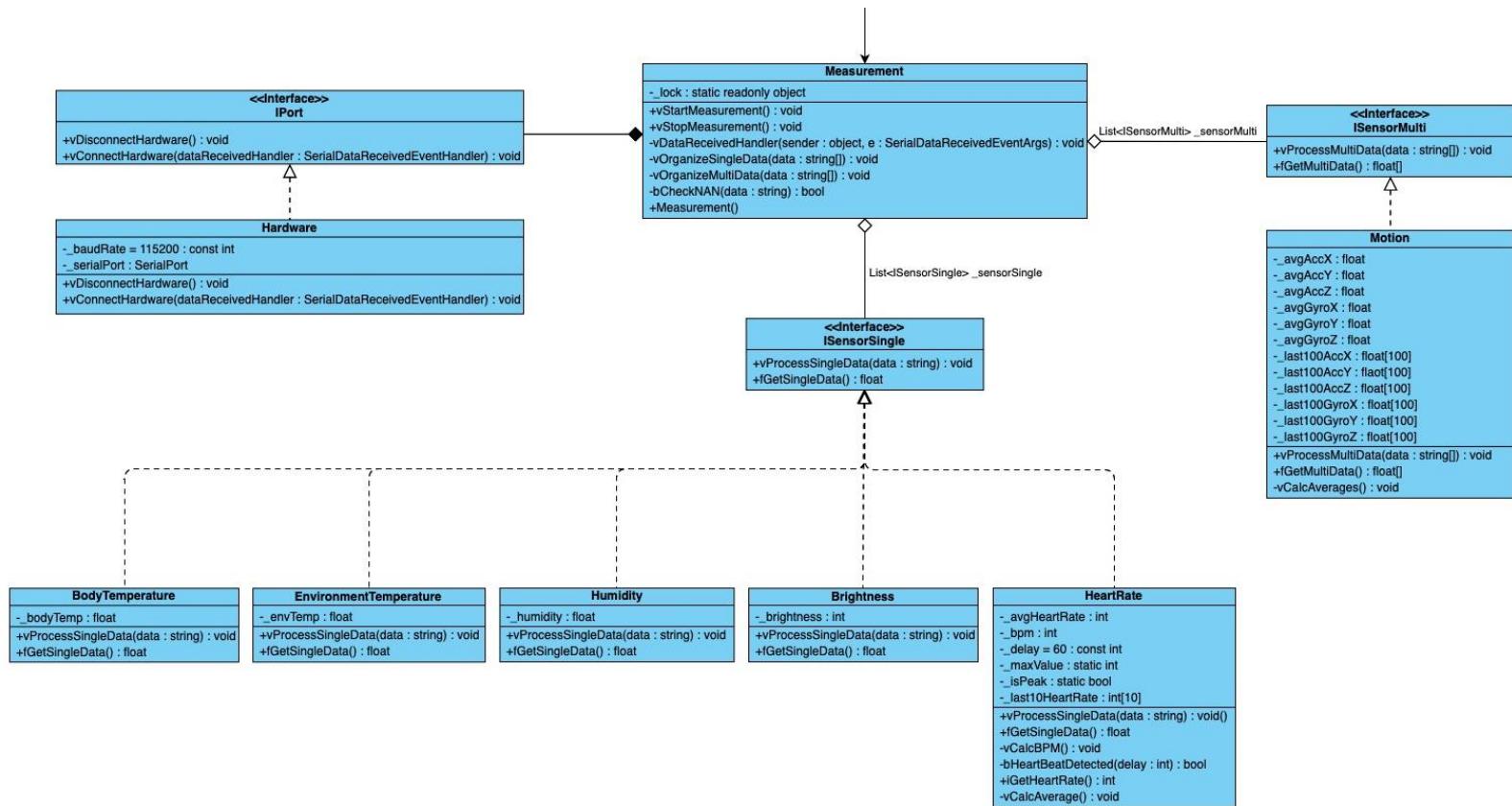


Abbildung 9 - UML-Klassendiagramm der Messungen

Um eine robuste Architektur zu gewährleisten, die durch zusätzliche Sensoren nach Bedarf erweitert werden kann, haben wir ein UML-Klassendiagramm entwickelt. Dieses Diagramm fördert durch die Nutzung von Interfaces eine konsistente Behandlung der Sensorklassen, was zu erhöhter Wartbarkeit und Erweiterbarkeit führt. Zudem erleichtert diese Struktur das Testen erheblich, da Mock-Implementierungen verwendet werden können, um das Verhalten der Sensoren zu simulieren, ohne dass physische Sensoren benötigt werden. Durch die Definition eines allgemeinen Hardware-Interfaces (`IPort`) wird die Integration unterschiedlicher Hardware-Komponenten stark vereinfacht. Neue Hardware-Typen können durch Implementierung dieses Interfaces problemlos hinzugefügt werden. Dies ermöglicht es, das System an sich ändernde Anforderungen oder neue Technologien anzupassen, ohne die bestehende Architektur grundlegend verändern zu müssen.

## Software Features

### Verbindung der API mit der Hardware:

Per HDMI-Kabel wird das Arduino Board an die Applikation angeschlossen. Die Hardware Klasse ist hier verantwortlich für die Kopplung an der COM-Schnittstelle und das empfangen der Daten. Eine leichte Erweiterung durch eine andere Hardware ist durch den Einsatz von dem interface IPort möglich. Die Hardware Klasse benötigt als Attribute daher nur die \_baudRate und den \_serialPort, um mit der API verbunden zu werden. Die Klassen werden in der nächsten Abbildung realisiert:

```
2 references
public interface IPort
{
    2 references
    void vDisconnectHardware();
    2 references
    void vConnectHardware(SerialDataReceivedEventHandler dataReceivedHandler);
}

1 reference
public class Hardware : IPort
{
    private const int _baudRate = 115200;
    private SerialPort _serialPort;
```

Abbildung 10 - Implementierung der Hardwareklasse

Die Methode vConnectHardware ermöglicht die Verbindung der Hardware mit der API. Durch die Übergabe eines Event Handlers für das DataReceived Event der seriellen Schnittstelle und der Baudrate \_baudRate wird ein neuer serieller Port erstellt und geöffnet. Der neue Port wird dann mit dem übergebenen Event Handler für das DataReceived Event registriert. Dies ermöglicht es, dass Daten, die über die serielle Schnittstelle empfangen werden, vom angegebenen Event Handler verarbeitet werden. Sollte der falsche Port angeschlossen sein oder gar kein Port angeschlossen sein, wird eine Exception mit einer entsprechenden Fehlermeldung ausgelöst. Die Methode wird in folgender Abbildung gezeigt:

```
/*
 * Call the function to connect the Serial Port
 */
2 references
public void vConnectHardware(SerialDataReceivedEventHandler dataReceivedHandler)
{
    _serialPort = new SerialPort("COM4", _baudRate);
    _serialPort.DataReceived += new SerialDataReceivedEventHandler(dataReceivedHandler);

    try
    {
        _serialPort.Open();
    }
    catch (Exception ex)
    {
        throw new Exception("Fehler beim Öffnen der seriellen Verbindung: " + ex.Message);
    }
}
```

Abbildung 11 - Implementierung der Verbindung zwischen API und Hardware

## Aufteilen der Daten mit der Measurement Klasse:

Nun übernimmt die Measurement Klasse. Die Methode DataReceivedHandler dient als Event-Handler, der alle über die serielle Schnittstelle empfangenen Sensordaten sammelt. Wenn Daten empfangen werden, wird diese Methode aufgerufen. Innerhalb der Methode wird zunächst ein lock auf das \_lock-Objekt gesetzt, um sicherzustellen, dass nur ein Thread zur gleichen Zeit auf den darin enthaltenen Code zugreifen kann, wodurch Thread-Sicherheitsprobleme vermieden werden.

Im lock-Block wird ein neuer Thread (workerThread) erstellt und gestartet. Dieser Thread führt den folgenden Code aus: Zuerst wird das SerialPort-Objekt (serialPort) aus dem sender-Parameter extrahiert. Es wird überprüft, ob der serielle Port geöffnet ist (serialPort.IsOpen). Wenn der Port geöffnet ist, liest der Code eine Zeile der empfangenen Daten (serialPort.ReadLine()) und entfernt führende und nachfolgende Leerzeichen (Trim()).

Die empfangenen Daten werden dann in einzelne Werte aufgeteilt, indem die Datenzeichenfolge an Kommas (',') getrennt wird (data.Split(',')). Die resultierenden Werte werden in einem Array (aData) gespeichert. Anschließend werden die Daten durch Aufrufen der Methoden vOrganizeSingleData und vOrganizeMultiData weiterverarbeitet.

Nach dem Starten des workerThread-Threads wird workerThread.Join() aufgerufen, um sicherzustellen, dass der Hauptthread wartet, bis der workerThread-Thread seine Arbeit abgeschlossen hat, bevor der lock-Block verlassen wird. Diese Schritte werden in der folgenden Abbildung dargestellt:

```

/*
 * Handler who collect all Sensor Datas
 */
1 reference
private void DataReceivedHandler(object sender, SerialDataReceivedEventArgs e)
{
    lock (_lock)
    {
        Thread workerThread = new Thread(() =>
        {
            SerialPort serialPort = (SerialPort)sender;
            if (serialPort.IsOpen)
            {
                //read Datas into string
                string data = serialPort.ReadLine().Trim();

                //seperate values
                string[] aData = data.Split(',');
                vOrganizeSingleData(aData);
                vOrganizeMultiData(aData);
            }
        });
        workerThread.Start();
        workerThread.Join();
    }
}

```

Abbildung 12 - Implementierung des Events um Daten zu übertragen

Die folgende Abbildung zeigt den Konstruktor der Klasse:

```

public Measurement()
{
    _sensorsSingle = new List<ISensorSingle>
    {
        new HeartRate(),
        new EnvironmentTemperature(),
        new Humidity(),
        new BodyTemperature(),
        new Brightness(),
    };
    _sensorsMulti = new List<ISensorMulti>
    {
        new Motion(),
    };
}

```

Abbildung 13 - Implementierung des Konstruktors

Die Klasse besitzt zwei Listen, welche einmal das Interface ISensorSingle und ISensorMulti als Objekt haben. ISensorSingle ist hierbei zuständig für die Sensoren, welche nur einen Messwert liefern. ISensorMulti ist zuständig für Sensoren, welche mehrere Meswerte zurückliefern.

```

public interface ISensorSingle
{
    void vProcessSingleData(string data);
    float fGetSingleData();
}

public interface ISensorMulti
{
    void vProcessMultiData(string[] data);
    float[] fGetMultiData();
}

```

Abbildung 14 - Implementierung der Interfaces

Sobald nun alle Werte in der Liste sind, wird die Form Klasse über ein Event informiert, dass sie die neuen Werte updaten kann.

### **Herzschlag:**

Der Herzschlag durchläuft einen Algorithmus, um den zugehörigen BPM zu berechnen und diesen auszugeben.

Um die aktuellen Messungen präzise darzustellen, haben wir uns entschieden, den Durchschnitt der letzten zehn Messungen zu berechnen und zu plotten. Dies ist notwendig, da der Sensor gelegentlich fehlerhafte Messwerte liefert. Durch die Verwendung des Durchschnittswerts wird das Diagramm zehnmal pro Sekunde mit neuen Werten aktualisiert, was zu einer verbesserten Stabilität des Diagramms führt. Der aktuelle Herzschlag wird so in Echtzeit im Diagramm angezeigt, wodurch der Nutzer stets eine verlässliche und aktuelle Darstellung erhält:



Abbildung 15 - Herzschlag Diagramm

Das Diagramm beinhaltet die Zeit auf der X-Achse, hier auf 5 Sekunden eingestellt und die gemessenen mV vom Sensor auf der Y-Achse.

Dazu erhält man eine Live BPM (Beat's per Minute) Anzeige, welche einem den genauen Herzschlag anzeigt.

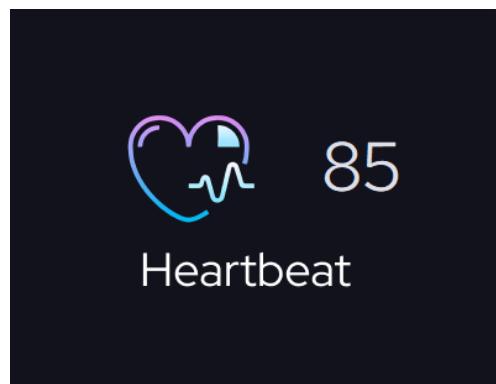


Abbildung 16 - Herzschlag (BPM)

Der Herzschlag wird weitergehend verwendet um über den Gesundheitszustand des Patienten informiert zu sein. Fällt der Herzschlag unter einen bestimmten Bereich oder steigt plötzlich gefährlich an, so wird eine Benachrichtigung an die angegebene Kontaktperson über die ausgewählte Variante (SMS/Email) gesendet.

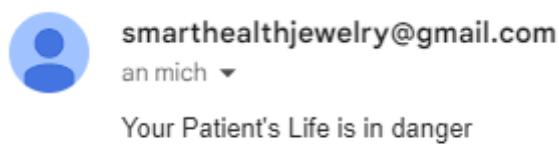
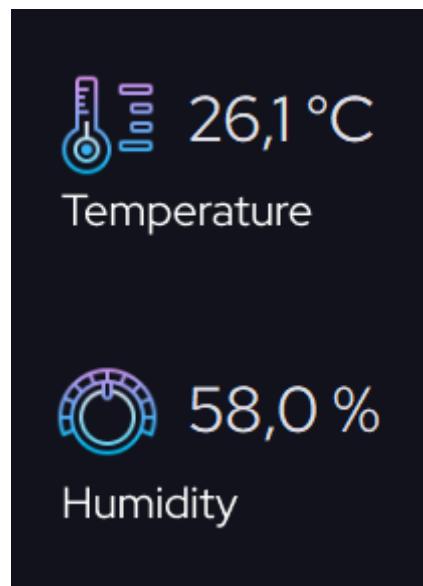


Abbildung 17 - Patient in Danger Email

Die Email wird im Moment noch an die Email-Adresse des Logins gesendet.

## **Feuchtigkeit und Umgebungstemperatur:**

Die Feuchtigkeitswerte dienen dazu, Informationen über die Umgebung der Person bereitzustellen. Bei zu hoher Feuchtigkeit kann der Nutzer über einen potenziell kritischen Gesundheitszustand informiert werden. Ebenso wird der Temperaturwert verwendet, um die Umgebungstemperatur der Person zu überwachen. Bei extremen Temperaturen, sei es zu warm oder zu kalt, kann der Nutzer ebenfalls über mögliche gesundheitliche Risiken benachrichtigt werden.



*Abbildung 18 - Umgebungsmesswerte*

## **Körpertemperatur:**

Die Überwachung der Körpertemperatur ist von entscheidender Bedeutung für die Gesundheitsüberwachung. Abweichungen von der normalen Körpertemperatur können auf verschiedene gesundheitliche Zustände hinweisen. Eine erhöhte Körpertemperatur, oft als Fieber bezeichnet, kann ein Indikator für Infektionen oder Entzündungen sein. Umgekehrt kann eine zu niedrige Körpertemperatur auf eine Hypothermie hinweisen, die ebenfalls dringend medizinische Aufmerksamkeit erfordert.

Durch die kontinuierliche Erfassung und Analyse der Körpertemperatur können frühzeitig Anomalien erkannt und der Nutzer rechtzeitig über potenzielle Gesundheitsprobleme informiert werden. Dies ermöglicht eine sofortige Reaktion und gegebenenfalls die Einleitung geeigneter medizinischer Maßnahmen, um die Gesundheit und das Wohlbefinden der Person zu schützen.

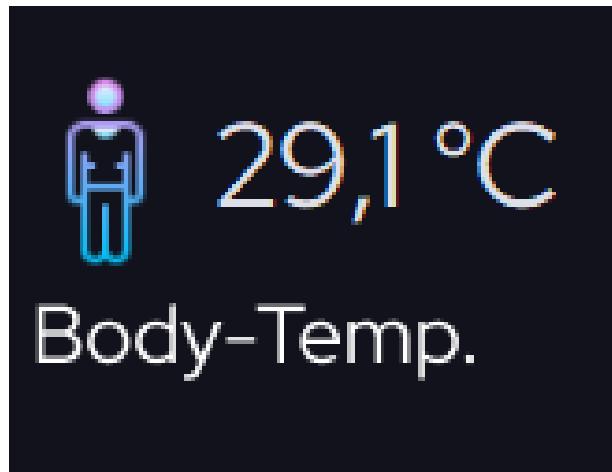


Abbildung 19 - Körpertemperatur

### Helligkeitserkennung:

Die Helligkeitserkennung erfolgt über den Helligkeitssensor. Der daraus resultierende ADC-Wert (zwischen 0 und 4095) wird zunächst durch die Converter-Klasse in einen Helligkeitsprozentsatz zwischen 0 und 100 % umgewandelt. Hierzu dient der folgende Code:

```
//GUI Color anpassen
private void AdjustBrightness(object sender, System.Timers.ElapsedEventArgs e)
{
    //Werte holen
    float brightnessVal = measurement._sensorsSingle[4].fGetSingleData();
    int BrightnessPercentage = 100 - (int)Math.Round(BackendCS.Converter.BrightnessConverter.convert(brightnessVal) * 100, 1);

    //wird gebraucht, damit GUI nicht komplett schwarz wird => 20 % Helligkeit braucht es schon
    //falls mehr sein soll muss nur der Wert geändert werden
    if (BrightnessPercentage < 20)
    {
        BrightnessPercentage = 20;
    }

    // Umrechnung des Prozentwerts in den RGB-Bereich (0-255)
    int brightnessValue = (int)(BrightnessPercentage * 2.55); // 255 / 100 = 2.55

    // Erstellen einer neuen Farbe mit den berechneten RGB-Werten
    Color backgroundColor = Color.FromArgb(brightnessValue, brightnessValue, brightnessValue);

    // Setzen der Hintergrundfarbe des Panels (oder des Formulars)
    this.BackColor = backgroundColor;
}
```

Abbildung 20 - Implementierung der automatischen Helligkeitsanpassung

Die Idee war es basierend auf diesem Prozentsatz alle 10 Sekunden die neue Helligkeit der GUI zu bestimmen. Je heller es draußen ist, desto heller muss auch die GUI sein, damit der Nutzer die Anzeige gut erkennen kann. Sobald das Umgebungslicht abnimmt, ist es augenschonend, wenn die Helligkeit der GUI entsprechend heruntergedimmt wird. Bei der Umsetzung kam es dann zu mehreren Problemen. Der Prototyp der GUI hatte das Feature erfolgreich implementiert, der jetzigen Endlösung fehlt dieses Feature aufgrund von Schnittstellenproblemen. Das Feature kann nachträglich noch implementiert werden, war jedoch aufgrund der Zeit und unseres Wissens nicht mehr unser Hauptziel.

## Bewegungsdaten:

Die empfangenen Beschleunigungs- und Winkelgeschwindigkeitswerte werden zunächst über einen Durchschnitt der letzten 100 Messungen geglättet, was einer Sekunde entspricht. Dieser Schritt dient dazu, Messfehler auszugleichen und eine präzise Darstellung der Bewegungsdaten zu gewährleisten.

Die verschiedenen Beschleunigungs- und Winkelgeschwindigkeitswerte werden dann in zwei separaten Diagrammen innerhalb der GUI angezeigt. Diese Diagramme sind spezifisch aufgeteilt in Bereiche für Beschleunigung und Winkelgeschwindigkeit entlang der X-, Y- und Z-Achsen. Die Auswahl, welche Achse angezeigt werden soll, erfolgt durch den Nutzer über Radiobuttons:



Abbildung 21 – Radiobuttons

Diese Aufteilung ermöglicht es dem Nutzer, die Bewegungsdaten in Echtzeit für jede Achse zu überwachen und mögliche Muster oder Abweichungen leicht zu erkennen. Durch die Anwendung des Durchschnitts der letzten 100 Messungen wird sichergestellt, dass die angezeigten Werte präzise und stabil sind, was für eine genaue Analyse der Bewegungsdaten essentiell ist.

In den folgenden Abbildungen werden sowohl Beschleunigungswerte als auch Winkelgeschwindigkeitswerte in der GUI dargestellt:

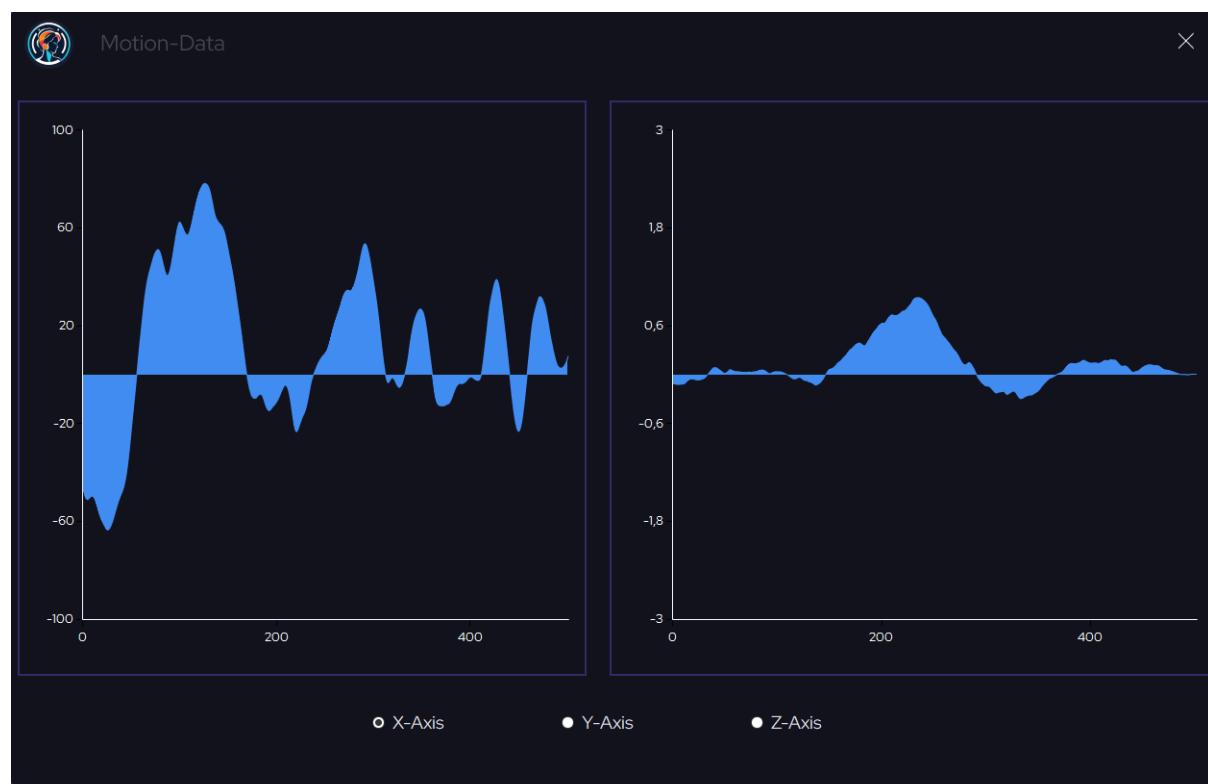


Abbildung 22 - X-Achse Bewegungsdaten



Abbildung 23 - Y-Achse Bewegungsdaten



Abbildung 24 - Z-Achse Bewegungsdaten

Die Diagramme bilden die letzten 5 Sekunden ab. Dem Nutzer ist die Möglichkeit gegeben über die Radiobuttons die verschiedenen Graphen anzuwählen.

## Zustand des Patienten:

Der Patient kann sich in verschiedenen Gesundheitszuständen befinden. Siehe Abbildungen.

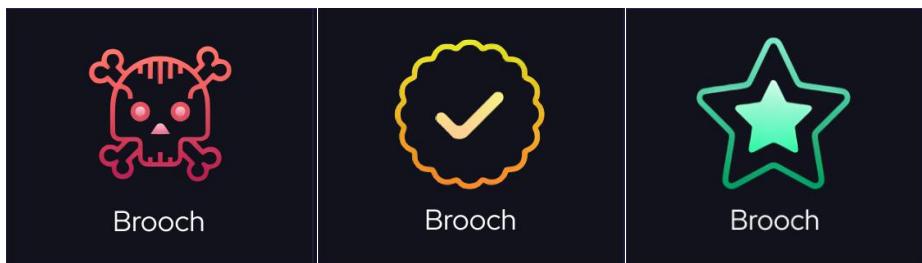


Abbildung 25 - Symbole für Gesundheitszustand

Der im Normalfall eintretende ist die grüne Broche. Dem Patienten geht es bestens, es besteht kein Grund zur Sorge. Das orangene Emblem zeigt an, dass dem Patienten etwas fehlt. Oftmals sind es nur kurzeitige Gesundheitsschwankungen, wie zum Beispiel eine erhöhte Außentemperatur. Sollte der Zustand jedoch länger anhalten ist Vorsicht geboten, es könnten Anzeichen eines ernsthaften Gesundheitsproblems sein. Wenn die Broche auf das rote Symbol umspringt ist schnelles Handeln gefragt. Der Patient ist einer echten Gefahrenlage. Hier wird auch über die später angesprochene Email-Methode direkt eine Email an die Kontaktperson versandt.

Aufgrund einiger Sensorprobleme und komplexen Algorithmen zu Berechnung des Gesundheitszustand lässt sich dieser über die drei Tasten „j“, „k“ und „l“ auf der Tastatur ändern.

## Kommunikation

Kommunikationsschnittstellen nach außen sind entscheidend für die Integration von Smart Health Jewelry in das Gesundheitssystem und die Interaktion mit externen Partnern. Diese Schnittstellen ermöglichen den sicheren Austausch von Gesundheitsdaten zwischen der Jewelry, medizinischen Einrichtungen, Betreuern und anderen relevanten Parteien.

Ein zentrales Element ist die bidirektionale Kommunikation zwischen der Jewelry und Gesundheitsdienstleistern. Über standardisierte Protokolle können Vitaldaten wie Herzschlag, Körpertemperatur und Blutzuckerwerte in Echtzeit an das medizinische Personal übertragen werden. Diese Daten sind entscheidend für die Fernüberwachung von Patienten und ermöglichen eine frühzeitige Intervention bei gesundheitlichen Problemen.

Ein weiterer wichtiger Aspekt sind die Schnittstellen zur elektronischen Patientenakte (EPA). Gesundheitsdaten, die von der Smart Health Jewelry erfasst werden, können nahtlos in die EPA

integriert werden. Dies ermöglicht eine konsolidierte Ansicht der Patientengesundheit über verschiedene medizinische Einrichtungen hinweg und erleichtert die Kontinuität der Versorgung. Ärzte können auf historische Daten zugreifen, Behandlungsverläufe überwachen und langfristige Gesundheitstrends analysieren, um fundierte medizinische Entscheidungen zu treffen. Hierbei handelt es um ein in Zukunft umsetzbares Feature.

Kommunikationsschnittstellen sind auch für die Interaktion mit Angehörigen und Pflegepersonen wichtig. Die Jewelry kann Informationen über den Gesundheitszustand des Trägers sicher an autorisierte Kontaktpersonen übermitteln. Dies fördert die Transparenz und ermöglicht es Familienmitgliedern, aktiv an der Unterstützung und Pflege ihrer Angehörigen teilzunehmen.

## E-Mail

Die Implementierung der Benachrichtigungsfunktionalität baut auf der Schnittstelle des NotificationMethods auf, welches durch SMS und E-Mail implementiert wird wie in Abbildung 1 zu sehen. Hierdurch wird die Erweiterbarkeit gewährleistet für andere Methoden, die später hinzugefügt werden können. Beispielsweise durch postalischen Versand oder App-Benachrichtigungen.

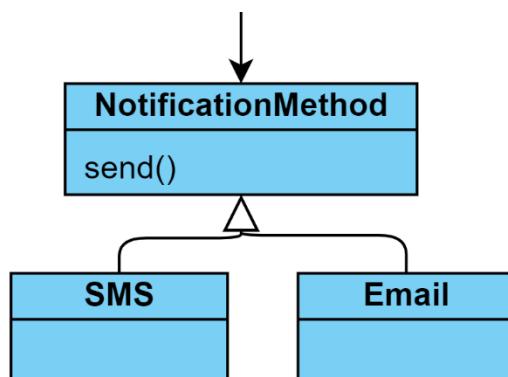


Abbildung 26 - Schnittstelle `NotificationMethod`

Für die beispielhafte Implementierung der Benachrichtigung per E-Mail wird von der Kontaktperson die E-Mail-Adresse verwendet, die durch die GUI übergeben werden kann. Anschließend kann vom User bestimmt werden auf welchem Weg, die Benachrichtigung erfolgen soll. Für die Applikation eigens wurde ein eigenes Google-Konto erstellt, um sicheren Versand per Mail zu gewährleisten. Durch Übergabe des Betreffs und Inhalt der Nachricht wird eine SMTP-Verbindung zum Gmail-Server aufgebaut und die Nachricht versendet. Gmail kümmert sich anschließend um die Zustellung der Nachricht nach erfolgreicher Authentifizierung mit einem App-Passwort. Abbildung 2 zeigt dabei den Sourcecode, um die eben beschriebene Funktionalität zu implementieren,

```
    public static void Send(string email, string subject, string message)
    {
        string mail = "smarthealthjewelry@gmail.com";
        string pw = "dvnz ufel zien mfvo";

        SmtpClient smtpClient = new SmtpClient(host: "smtp.gmail.com")
        {
            Port = 587,
            Credentials = new NetworkCredential(userName: mail, password: pw),
            EnableSsl = true,
        };

        MailMessage mailMessage = new MailMessage
        {
            From = new MailAddress(mail),
            Subject = subject,
            Body = message
        };

        mailMessage.To.Add(email);

        smtpClient.Send(mailMessage);

        Console.WriteLine("sending was successful");
    }
}
```

Abbildung 27 - Implementierung E-Mail Benachrichtigung

# Erweiterung der Brosche durch GUI-Applikation

## 1) Vorstellung

Eine GUI-Applikation soll Ärzten / Pflegern / Familienmitgliedern zusätzliche Informationen sowie Möglichkeiten geben, die aktuelle Gesundheitslage ihrer Familienmitglieder besser einschätzen sowie überschauen zu können. Die Applikation soll damit die Funktionalitäten und den Einsatzbereich der Brosche erweitern, damit auch digital von anderen Orten gearbeitet werden kann, ohne im selben Raum wie der Patient sein zu müssen.

## 2) Design der GUI-Desktop Applikation

Das Design wurde mittels der kostenlosen Webanwendung von FIGMA erstellt und gewartet.

Im ersten Schritt wurden verschiedene bereits existierende GUI-Designs gesammelt, um eine Inspirationsquelle für die Smart-Health-Desktop Anwendung zu erstellen. Manche dieser Bilder wurden sogar per KI (s. Github Readme) generiert, um eine möglichst genaue Vorstellung für das eigene Produkt erhalten zu können.

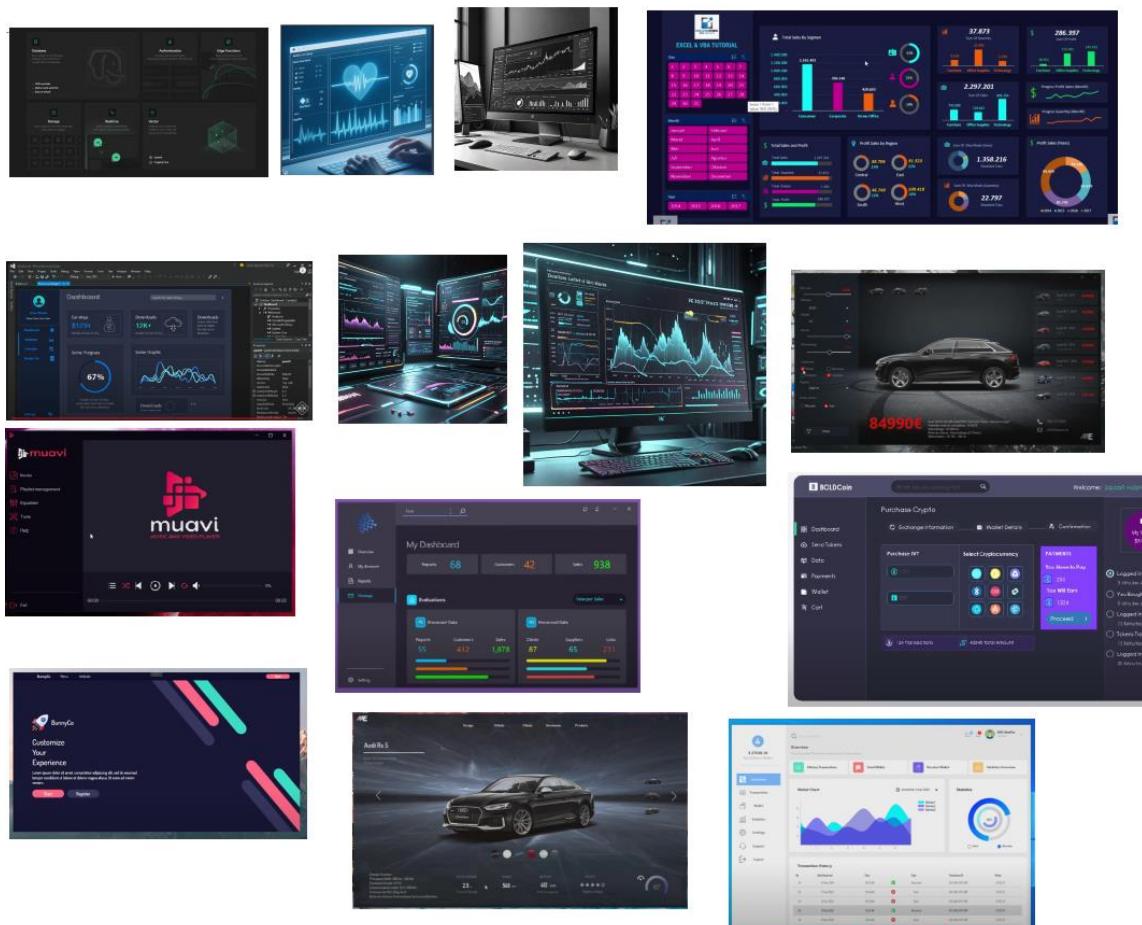


Abbildung 28 - Erstellung einer Inspirationsquelle (Ausschnitt) für einen Main-Screen

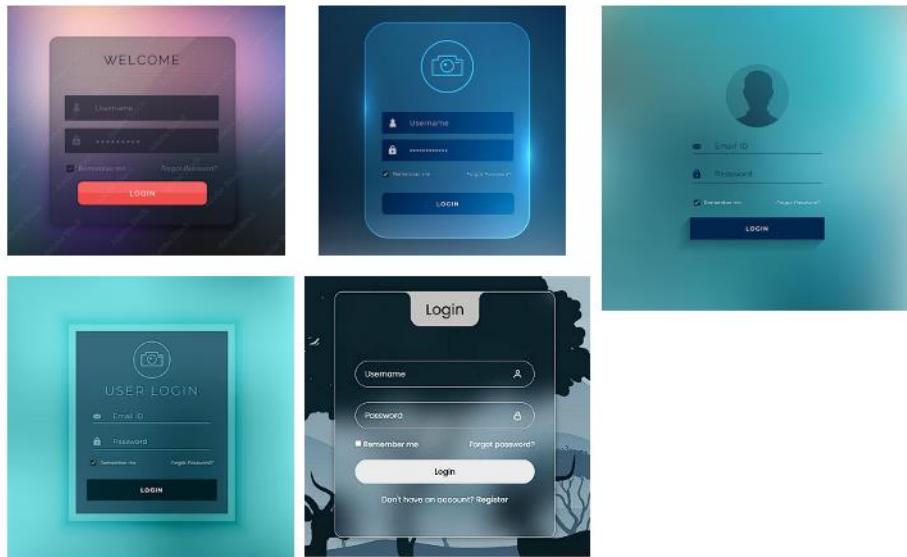


Abbildung 29 - Inspiration Login-Screen

Auf dieser Grundbasis aufbauend wurde der Arbeitsflow für das Programm festgelegt, um eine klare Struktur zu bekommen, welche verschiedenen Fenster die App haben muss und in welcher Reihenfolge diese zu öffnen sind.

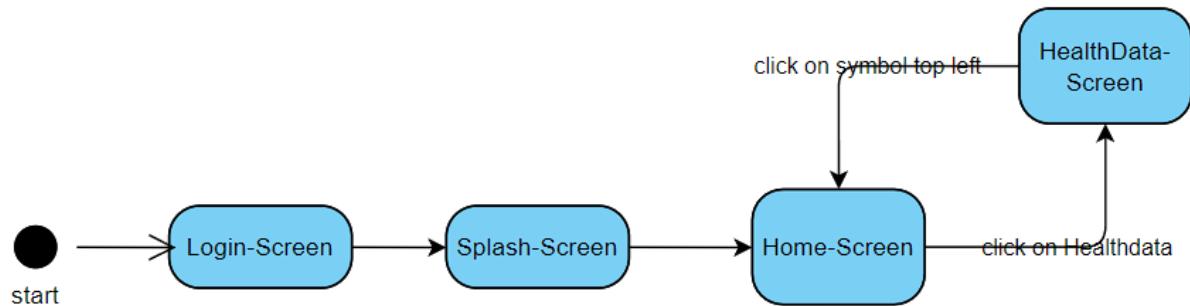


Abbildung 30 - Programmfluss-Darstellung

Mithilfe des fertigen Programmflusses wurden die Designs der einzelnen Fenster der Anwendung wieder mit FIGMA erstellt. Zuerst wurden die Hauptelemente bestimmt und geordnet, um einen Grobentwurf (Mockup) zu bekommen.

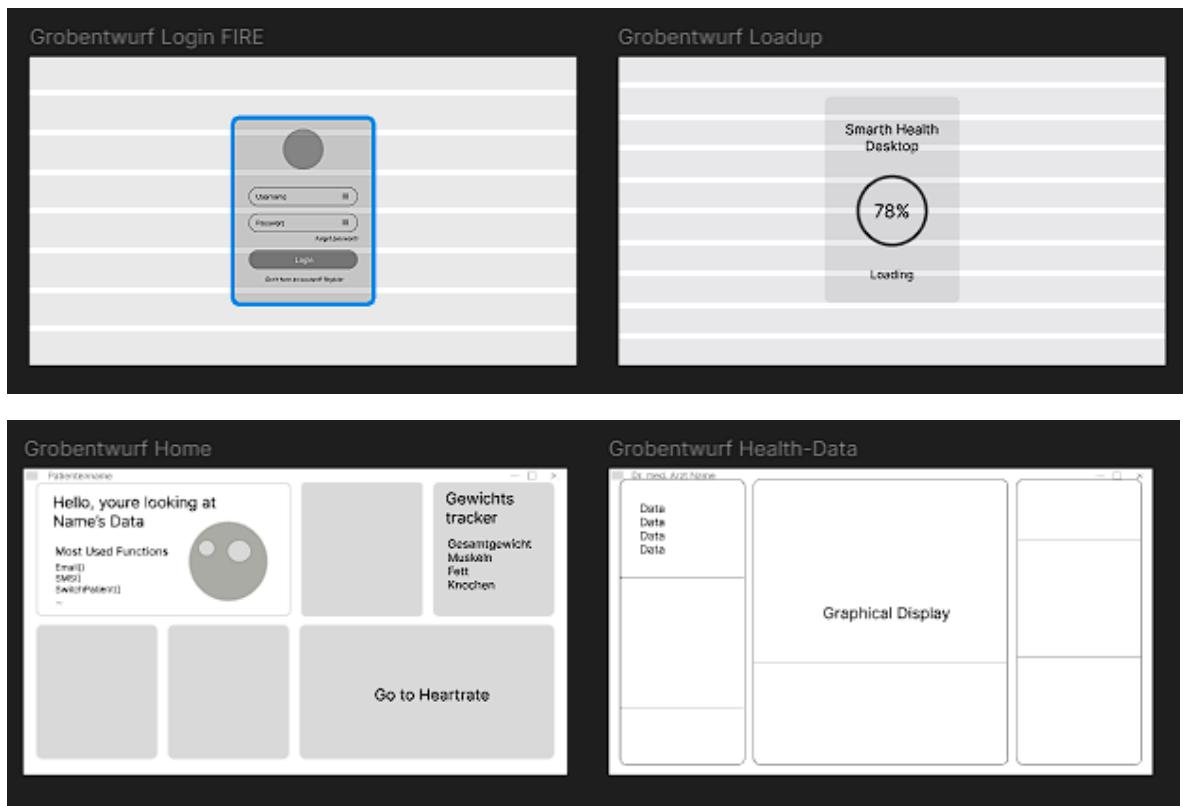


Abbildung 31 - Ausschnitt einzelner Mockups der GUI

Für jedes dieser Fenster konnte anschließend das Design finalisiert werden, in dem verschiedene Endresultate erstellt wurden, aus denen jeweils eins umgesetzt wurde.

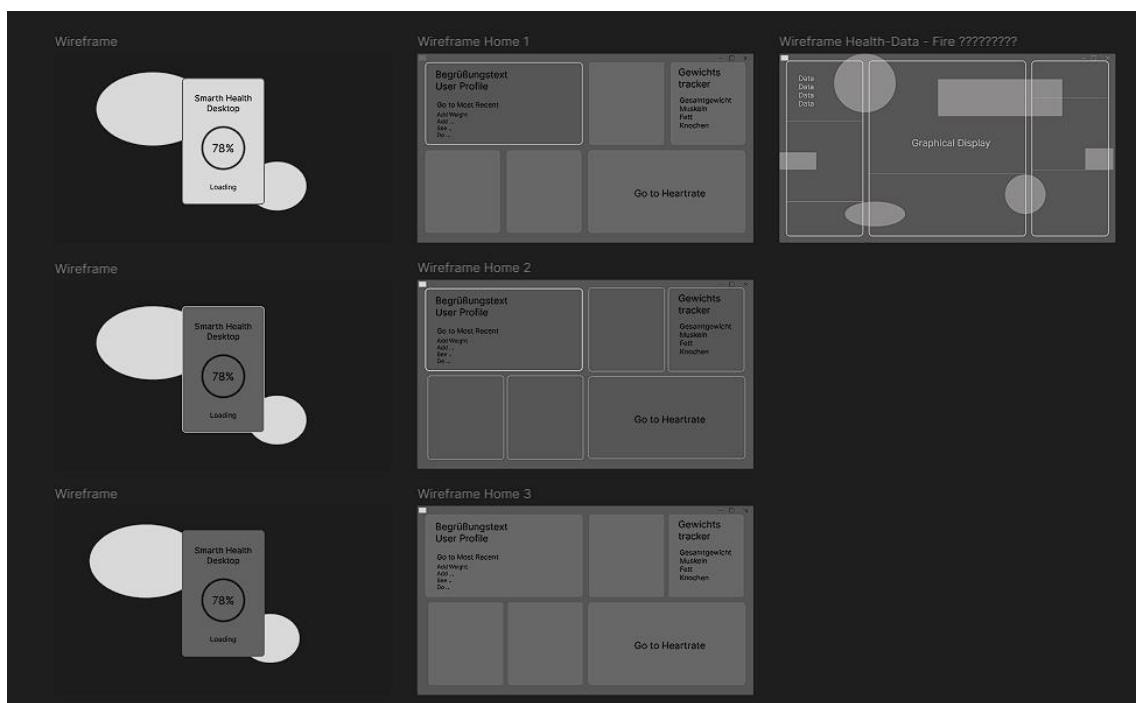


Abbildung 32 - Ausschnitt verschiedener Designentwürfe der GUI-Fenster

Parallel zum Designablauf der GUI durchlief das Design der Farben ähnliche Schritte und folgte demselben Schema/Reihenfolge wie oben beschrieben.

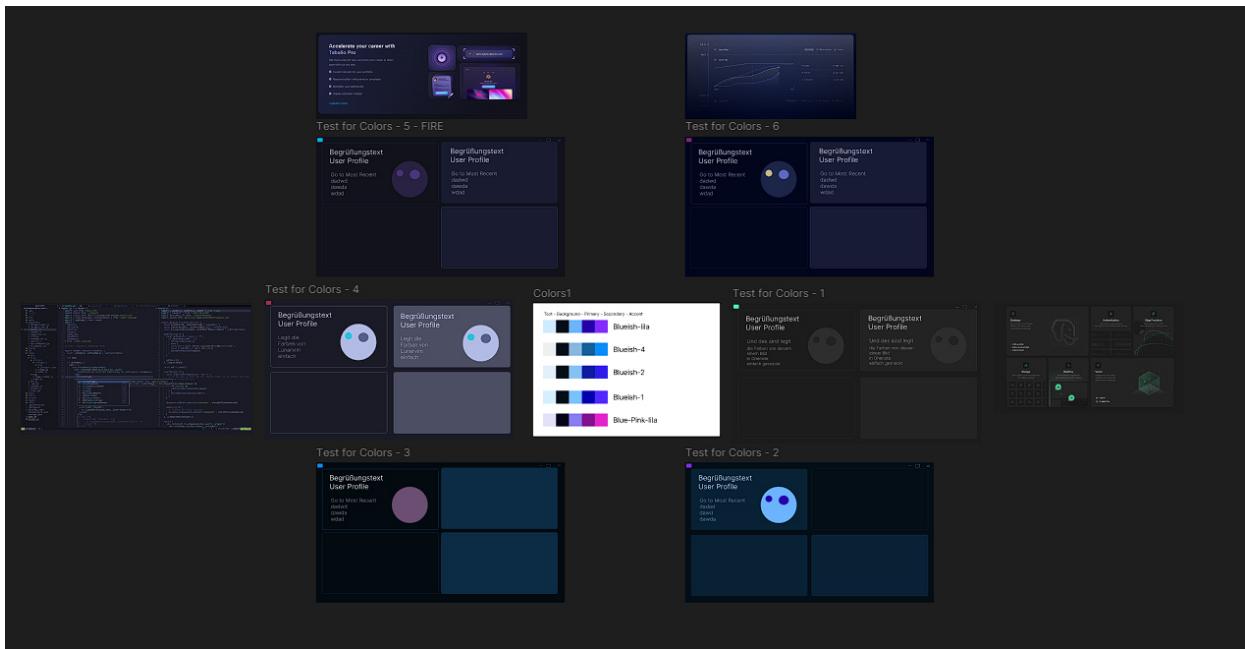


Abbildung 33 - Design der Farben

### 3) Erstellung / Programmierung des Designs

Das Design wurde in Microsoft Visual Studio Code (MCVS) per Windows-Forms realisiert. Im Projekt User-Interface dieser Solution sind folgende Klassen daher das Hauptaugenmerk.

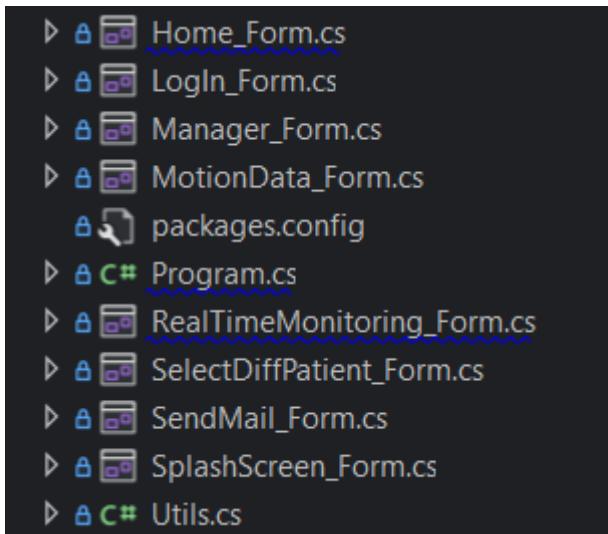
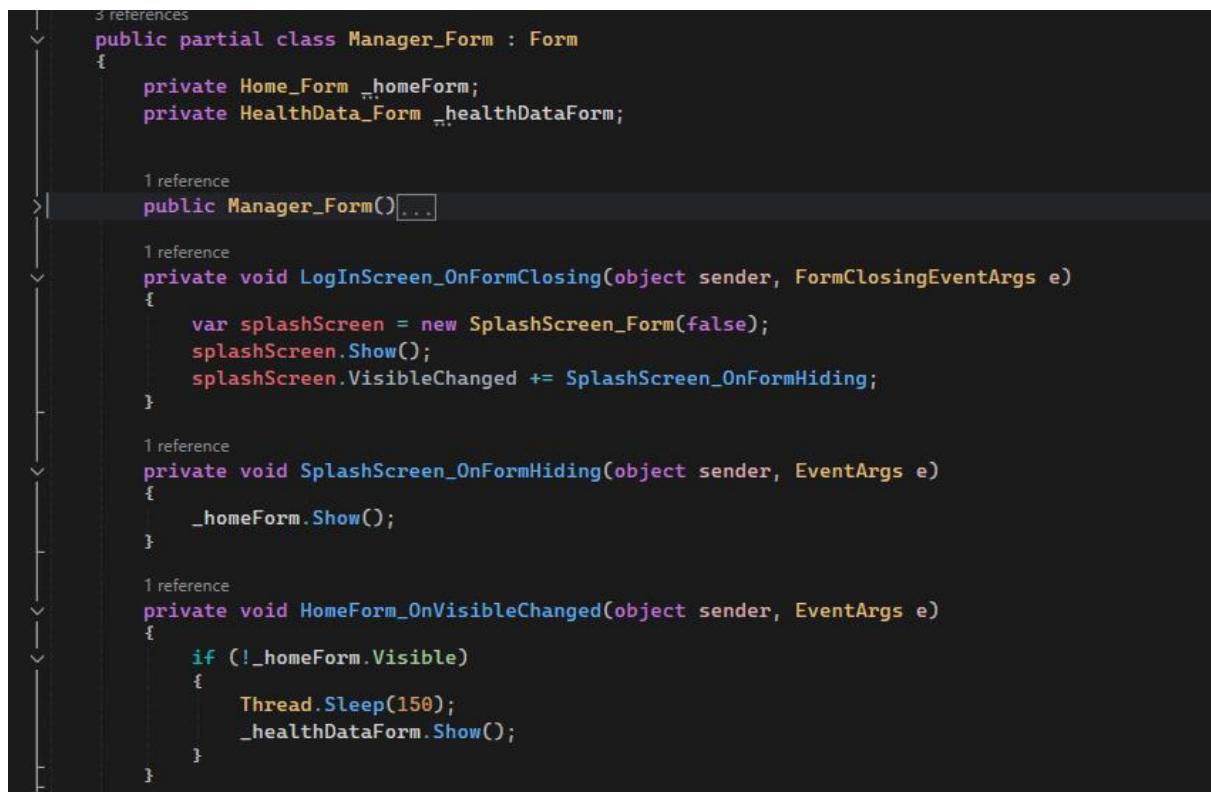


Abbildung 34 - Ausschnitt der Klassen des User-Interfaces

Jedes Fenster der GUI ist ein eigenes Windows-Form Element. Zentrale Rolle spielt hierbei das Manager-Form, das für die Verwaltung aller Forms (=Screens) zuständig ist. Die verschiedenen Forms werden in dieser Klasse erstellt und auf deren Events registriert, um z.B. beim Schließen des Login-Screens nach erfolgreicher Anmeldung zunächst den Lade-Screen und danach den Home-Screen anzuzeigen. Hierfür wurden die von Microsoft bereits zur Verfügung gestellten Events wie „FormClosing“ oder „VisibleChanged“ verwendet.



```

3 references
public partial class Manager_Form : Form
{
    private Home_Form _homeForm;
    private HealthData_Form _healthDataForm;

    1 reference
    public Manager_Form()...

    1 reference
    private void LogInScreen_OnFormClosing(object sender, FormClosingEventArgs e)
    {
        var splashScreen = new SplashScreen_Form(false);
        splashScreen.Show();
        splashScreen.VisibleChanged += SplashScreen_OnFormHiding;
    }

    1 reference
    private void SplashScreen_OnFormHiding(object sender, EventArgs e)
    {
        _homeForm.Show();
    }

    1 reference
    private void HomeForm_OnVisibleChanged(object sender, EventArgs e)
    {
        if (!_homeForm.Visible)
        {
            Thread.Sleep(150);
            _healthDataForm.Show();
        }
    }
}

```

Abbildung 35 - Ausschnitt der Manager-Form Klasse

Die mit FIGMA vorher erstellten Design-Schablonen konnte leicht auf die Klassen übertragen werden, indem die Design-Funktionalitäten der verwendeten IDE Microsoft-VS genutzt wurden (Abbildung 36). Um die Probleme und Schwierigkeiten, schöne Designs mit Windows-Forms erstellen zu können, zu umgehen, wurde der Großteil in FIGMA erstellt, und dieses Design als Hintergrund für jedes GUI-Fenster verwendet. Dadurch mussten nur noch Buttons und Labels von Win-Forms verwendet werden, was die Umsetzung des Designs um ein vielfaches erleichterte.

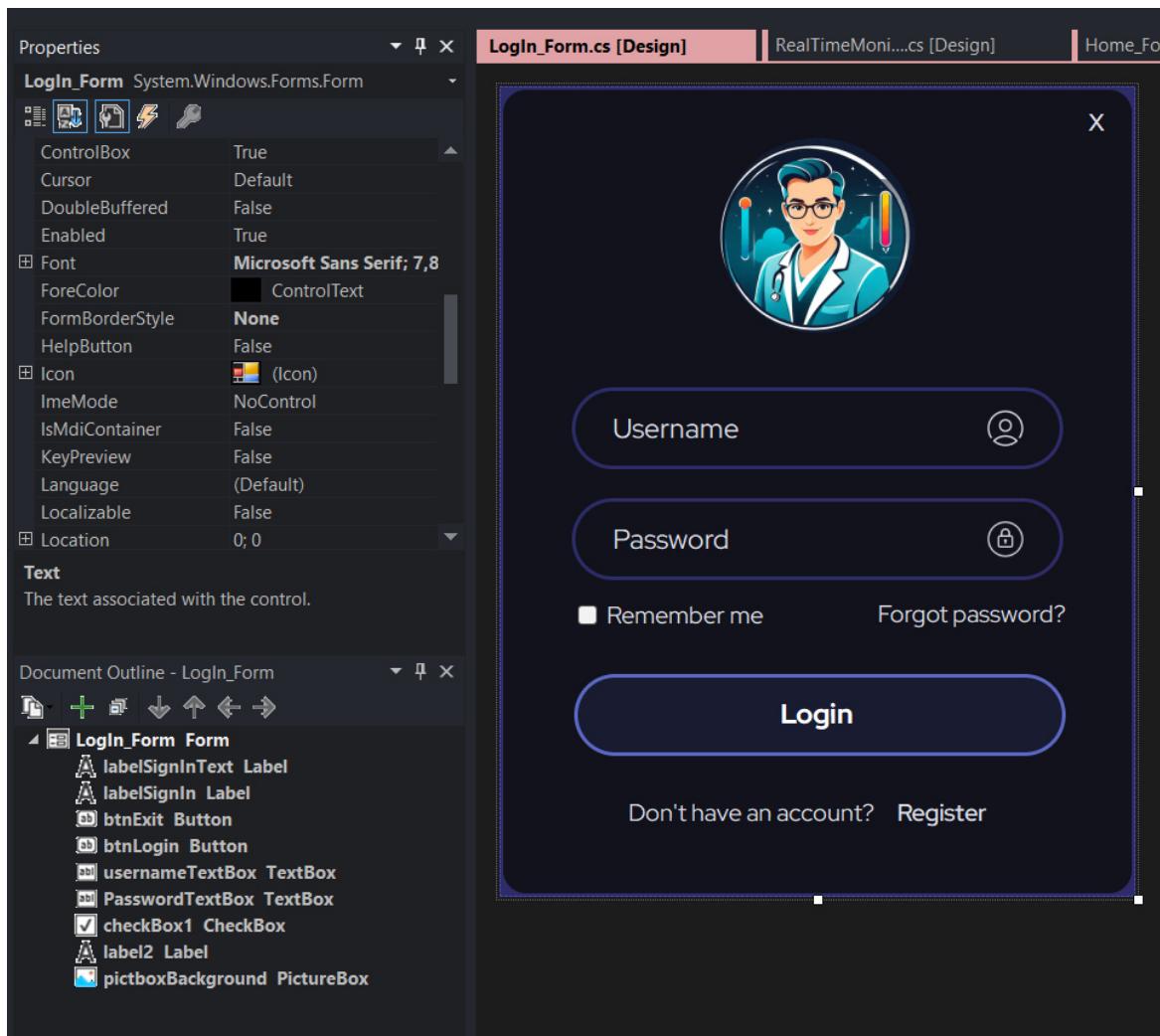


Abbildung 36 - Übertragen des Design aus FIGMA auf die einzelnen Klassen

## 4) Testen einzelner Screens

Damit das Testen einzelner Komponenten (Screens) bzw. deren Design leicht zu handhaben war, wurde in dem Starterfile des Programms die Funktionalität hinzugefügt, spezifizieren zu können, in welcher Laufumgebung das Programm sich befinden soll, z.B. Normaler Programmablauf oder nur starten des Login-Screens.

```
7 references
enum RunSettings
{
    Normal = 0,
    SplashScreen_Only, LogIn_Only, Home_Only, HealthData_Only,
    Skip2Home      // todo
};

0 references
internal static class Program
{
    // ----- Start CONFIG : Free to use -----
    static RunSettings CurRunSettings = RunSettings.HealthData_Only;
    // ----- END CONFIG -----

    /// <summary> The main entry point for the application.
    [STAThread]
    0 references
    static void Main()
    {
        Application.EnableVisualStyles();
        Application.SetCompatibleTextRenderingDefault(false);

        Form StartForm = null;
        switch(CurRunSettings)
        {
            case RunSettings.Normal:
                StartForm = new Manager_Form();
                break;

            case RunSettings.SplashScreen_Only:
                StartForm = new SplashScreen_Form(true);
                break;

            case RunSettings.LogIn_Only:
                StartForm = new LogIn_Form();
                break;

            case RunSettings.Home_Only:
                StartForm = new Home_Form(false);
                break;

            case RunSettings.HealthData_Only:
                StartForm = new HealthData_Form(false);
                break;

            default:
                throw new Exception("error: this shouldnt happen");
        }
        Application.Run(StartForm);
    }
}
```

Abbildung 37 - Kurzfassung des Starterfiles „Programm.cs“ zum Anpassen der Startkonfiguration

## 5) GUI

### Login:

Das Loginfenster wird geöffnet, sobald der Nutzer die Applikation startet.

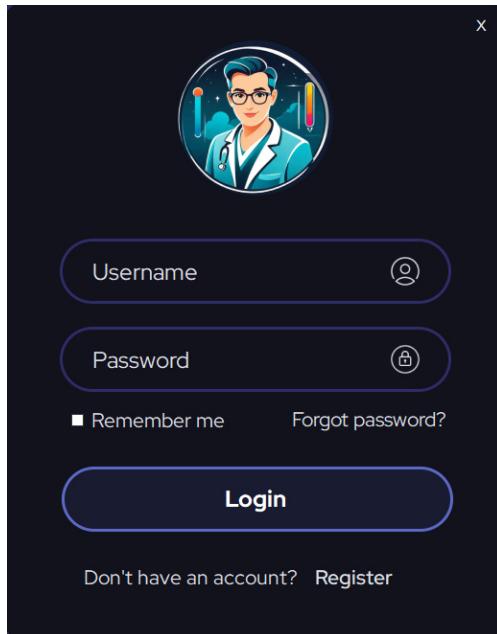


Abbildung 38 - Login

Als Nutzernname sind lediglich echte Email-Adressen möglich. Falsche Eingaben und leere Eingaben werden auf dem entsprechenden Label angezeigt.

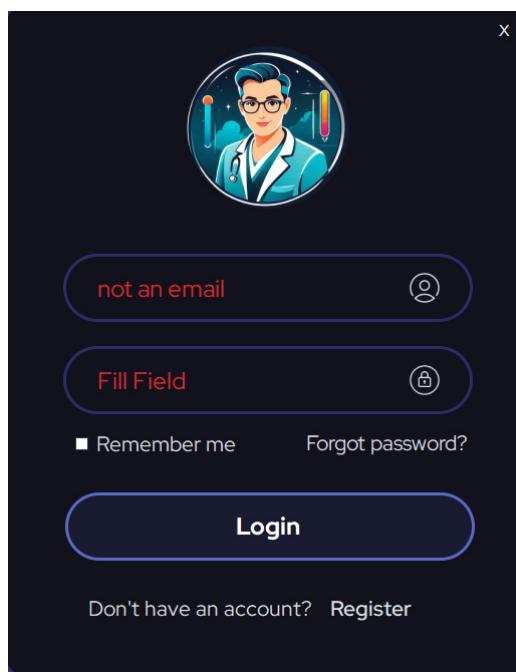


Abbildung 39 - Login: falsche Werte

Es ist nicht möglich mit einer Email-Adresse mehrere Accounts zu erstellen.

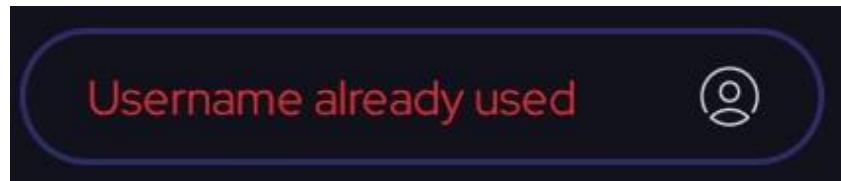


Abbildung 40 – Register: Email schon vergeben

Der Login funktioniert nur wenn es diesen Account schon gibt. Sprich es musste sich einmal Registriert werden.

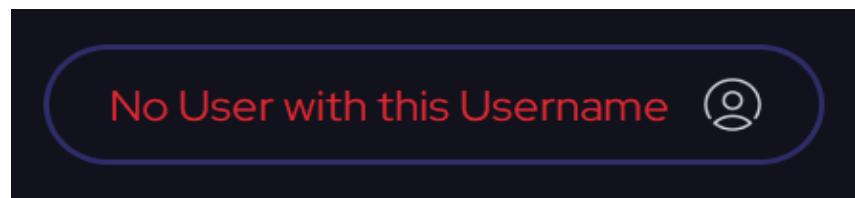


Abbildung 41 - Register: Email nicht vorhanden

Der Login funktioniert nur mit dem richtigen Passwort.

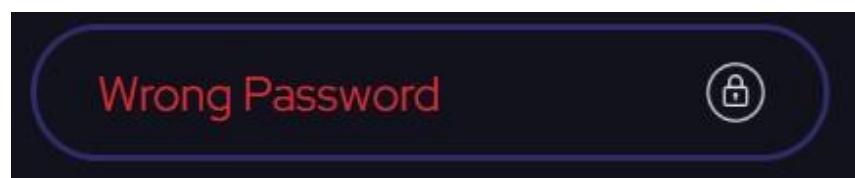


Abbildung 42 - Login: falsches Passwort

Nach einem erfolgreichen Login kommt man zum Load-Up Screen. Nach dem fertigen Load-Up landet man im Home-Screen.

## Home-Screen:

Der Home-Screen bietet mehrere Fenster, welche auf Nutzereingaben agieren.

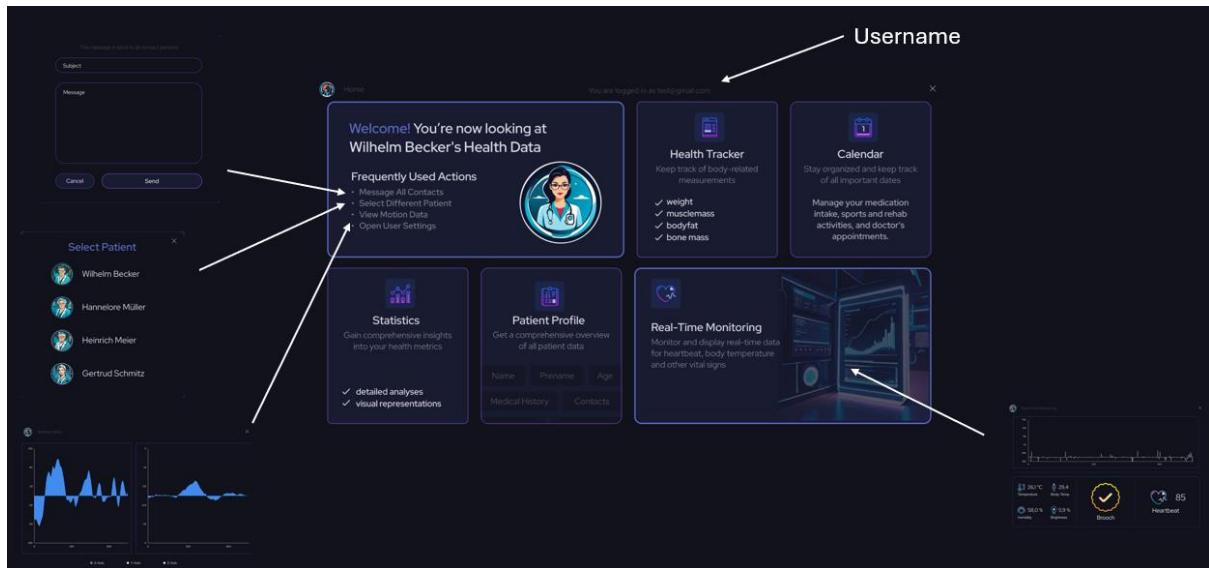


Abbildung 43 - Home Screen: detaillierte Bereiche

## Message All:

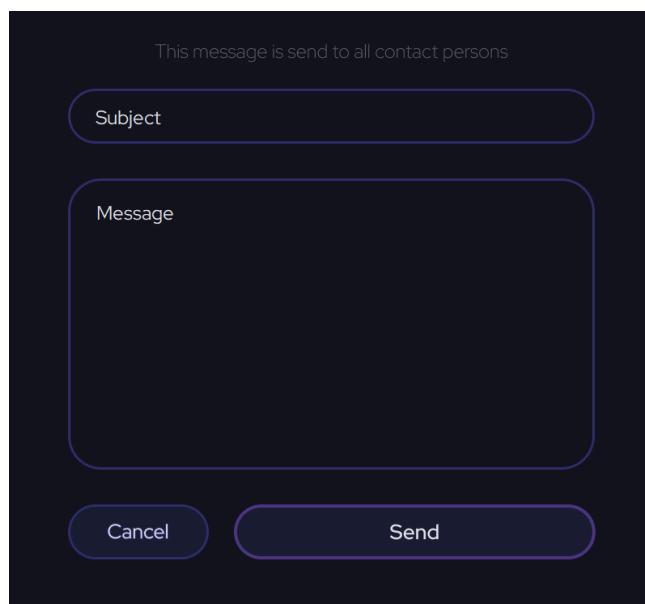


Abbildung 44 - message all window

Benachrichtige die Kontaktpersonen per Hand.

## Select Patient:

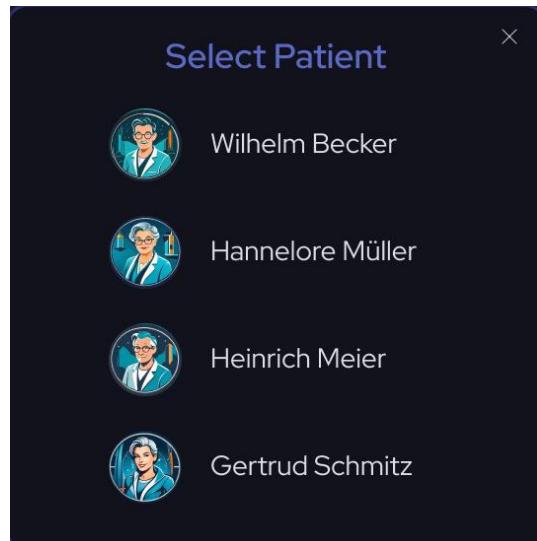


Abbildung 45 - select patient window

Wähle einen schon erstellten Patienten, mit speziellen Datensätzen.

## Motion-Data:



Abbildung 46 - motion window

Die Möglichkeiten dieses Fensters wurden schon im weiter vorne in der Dokumentation detailliert angesprochen. In diesem Fenster sieht der Nutzer seine Realtime Bewegungsdaten.

## Realtime-Monitoring:

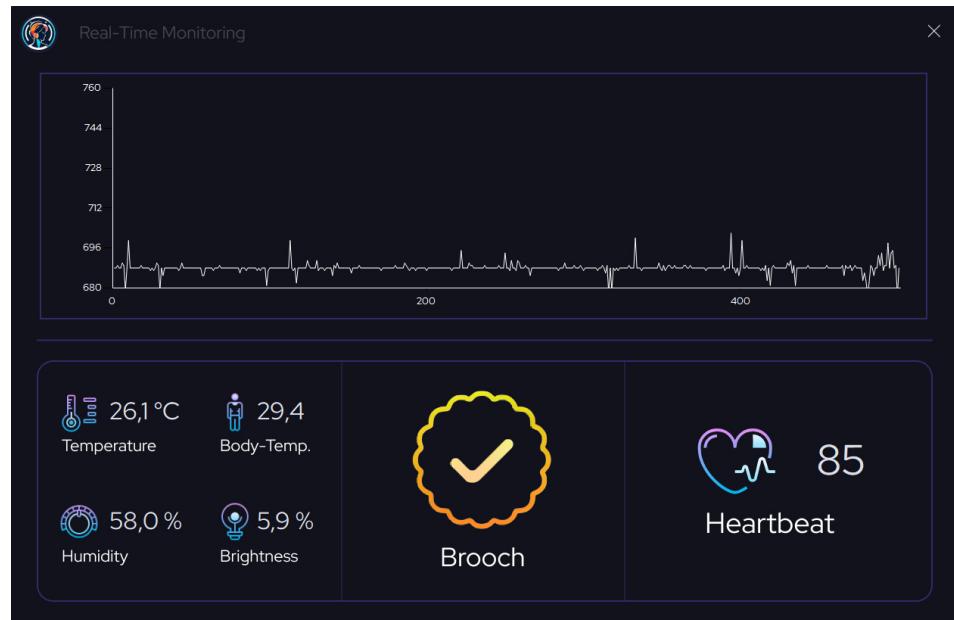


Abbildung 47 - realtime monitoring window

Auch dieses Fensters wurden schon im weiter vorne in der Dokumentation detailliert angesprochen. Hier kann der Nutzer seinen Herzschlag und seine allgemeine Gesundheitsverfassung einsehen.

## Daten

Im Rahmen des Projekts zur Entwicklung einer Smart Health Jewerly, die speziell für ältere Senioren konzipiert ist, wurden zahlreiche Funktionen integriert, die umfassende Gesundheitsdaten erfassen und auswerten. Die Anwendung speichert Daten zu wichtigen Kontaktpersonen, wie Familienmitgliedern, Freunden oder medizinischen Betreuern, wie in Abbildung 1 zu sehen.

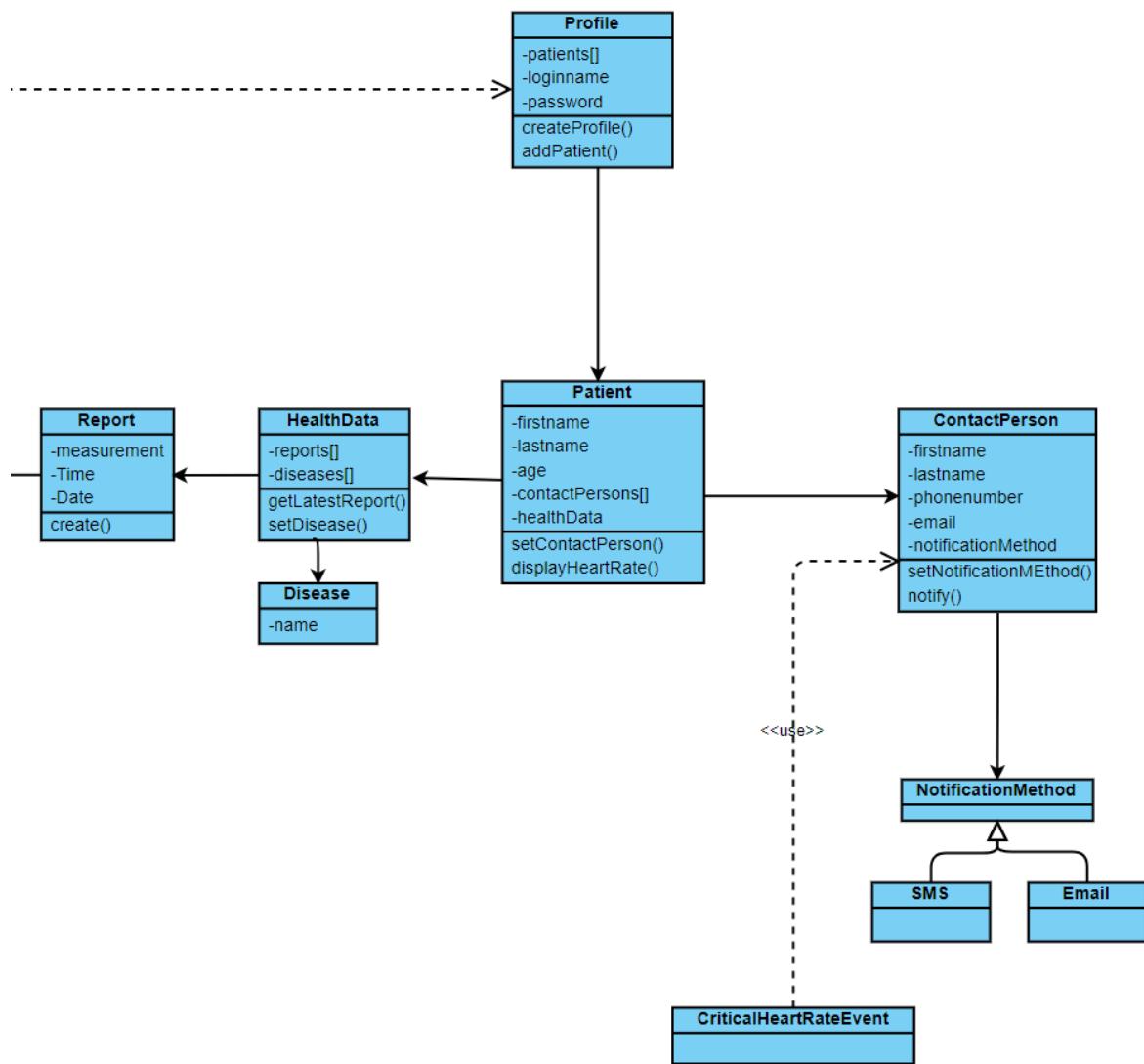


Abbildung 48 - Architektur Patiendatenverwaltung

In Notfällen kann so, eine Benachrichtigung an die hinterlegten Kontaktpersonen gesendet werden, was eine schnelle Hilfe ermöglicht. Über die Applikation können diese Personen auch direkt kontaktiert werden. Informationen über bestehende Krankheiten wie Diabetes,

Bluthochdruck oder Herzkrankheiten werden ebenfalls erfasst. Diese Daten ermöglichen eine personalisierte Gesundheitsüberwachung und helfen, Anomalien frühzeitig zu erkennen.

Die Jewelry misst kontinuierlich den Herzschlag durch optische Sensoren, die die Blutflussgeschwindigkeit unter der Haut erfassen. Durch die Überwachung des Herzschlags können Unregelmäßigkeiten wie Arrhythmien erkannt werden. Auffällige Herzrhythmen führen zu sofortigen Benachrichtigungen, die eine medizinische Intervention ermöglichen. Die Hautfeuchtigkeit wird ebenfalls überwacht, was Rückschlüsse auf den Hydratationsstatus des Körpers zulässt. Besonders für Senioren ist ausreichende Hydratation, vor allem im Sommer, wichtig. Bei niedrigen Feuchtigkeitswerten erinnert die Jewelry den Nutzer daran, genügend zu trinken, um Dehydrierung zu vermeiden.

Die Körpertemperatur wird regelmäßig gemessen, um Fieber oder Unterkühlung zu erkennen. Eine erhöhte Temperatur kann auf eine Infektion hinweisen, während eine niedrige Temperatur auf Unterkühlung hindeutet. Zusätzlich ist ein Sturzsensor integriert, der Stürze erkennt und automatisch eine Benachrichtigung der Kontaktpersonen oder Rettungsdienste sendet. Dies erhöht die Sicherheit der Träger erheblich, da schnelle Hilfe gewährleistet ist, selbst wenn der Betroffene nicht in der Lage ist, selbstständig Hilfe zu rufen.

Um den Nutzen der Smart Health Jewelry weiter zu erhöhen, ist es vorgesehen einen Raum für Erweiterungen zu belassen. Beispielsweise könnten Erweiterungen wie die Messung der Sauerstoffsättigung im Blut sinnvoll sein. Dies könnte zur frühzeitigen Erkennung von Atemwegserkrankungen beitragen. Ein Erinnerungssystem zur rechtzeitigen Einnahme von Medikamenten sowie eine GPS-Funktion zur Lokalisierung bei Desorientierung oder im Notfall wären ebenfalls nützliche Ergänzungen.

## Datenweiterverarbeitung

Die Integration dieser Daten in eine elektronische Patientenakte (EPA) bietet Ärzten ein umfassendes und kontinuierliches Gesundheitsbild eines Patienten. Dies ermöglicht eine genauere Diagnose und individuell angepasste Behandlung. Beispielsweise können Trends und Muster in den Vitaldaten erkannt werden, die auf sich entwickelnde Gesundheitsprobleme hinweisen. Die kontinuierliche Überwachung und Datenaufzeichnung erleichtert zudem die Nachverfolgung von Behandlungserfolgen und Anpassungen von Therapien.

Die Datenweiterverarbeitung bietet auch die Möglichkeit der Integration von Algorithmen und künstlicher Intelligenz. Solche Technologien können große Datenmengen analysieren und Muster erkennen, die menschlichen Ärzten möglicherweise entgehen. Beispielsweise könnte ein Algorithmus frühzeitig Anzeichen für Herz-Kreislauf-Probleme oder Stoffwechselstörungen erkennen und Warnungen ausgeben, bevor klinische Symptome auftreten. Diese prädiktive Analyse könnte die Prävention und frühzeitige Behandlung von Krankheiten revolutionieren.

Für Patienten bedeutet die Nutzung einer elektronischen Patientenakte mehr Transparenz und Kontrolle über ihre Gesundheitsdaten. Sie können ihre eigenen Daten einsehen und aktiv an ihrer Gesundheitsvorsorge teilnehmen, indem sie beispielsweise ihre Vitalwerte überwachen und an telemedizinischen Konsultationen teilnehmen. Dies stärkt die Patientensouveränität und fördert eine partnerschaftliche Beziehung zwischen Arzt und Patient.

## Datenschutzkonformität

Ein zentraler Aspekt bei der Nutzung von Gesundheitsdaten ist die Einhaltung der Datenschutzbestimmungen. Die Speicherung und Verarbeitung von Gesundheitsdaten muss gemäß den geltenden Datenschutzgesetzen, wie der Datenschutz-Grundverordnung (DSGVO), erfolgen. Dies beinhalten die Anonymisierung und Verschlüsselung der Daten, um die Privatsphäre der Patienten zu schützen. Zugriffsrechte müssen streng kontrolliert und nur autorisierten Personen gewährt werden. Zudem sollten Patienten transparent über die Verwendung ihrer Daten informiert werden und die Möglichkeit haben, der Datenverarbeitung zu widersprechen.



Abbildung 49 – Datenschutz: Grundverordnung

## Ethische Bedenken

Neben den Datenschutzaspekten gibt es auch ethische Bedenken bei der Nutzung und Verarbeitung von Gesundheitsdaten. Ein zentrales ethisches Anliegen ist die Wahrung der Autonomie der Patienten. Patienten müssen vollständig informiert sein und ihre Zustimmung zur Datenerfassung und -verarbeitung geben. Diese Zustimmung sollte freiwillig und ohne Druck erfolgen, um die Entscheidungsfreiheit der Patienten zu gewährleisten.

Ein weiteres ethisches Problem ist das Risiko des Datenmissbrauchs. Es besteht die Gefahr, dass Gesundheitsdaten für unethische Zwecke verwendet werden, etwa zur Diskriminierung in

der Versicherung oder am Arbeitsplatz. Es ist daher unerlässlich, robuste Sicherheitsmaßnahmen zu implementieren und sicherzustellen, dass die Daten nur zu legitimen medizinischen Zwecken verwendet werden.

Zudem muss die Frage der Gerechtigkeit berücksichtigt werden. Der Zugang zu Technologien wie der Smart Health Jewelry und der elektronischen Patientenakte darf nicht zu einer Zwei-Klassen-Medizin führen, bei der nur wohlhabende oder technologisch versierte Patienten von den Vorteilen profitieren. Es ist wichtig, sicherzustellen, dass diese Technologien allen Patienten zugänglich gemacht werden, unabhängig von ihrem sozioökonomischen Status.

## Schluss

### Rückblick auf das Projekt

Im Rahmen unseres Projekts zur Entwicklung der Smart Health Jewelry Desktop Applikation haben sich mehrere Aspekte als besonders erfolgreich erwiesen. Unsere Ideenphase war geprägt von kreativen Designvorschlägen, die die Grundlage für unsere spätere Umsetzung bildeten. Über den Gestaltungsprozess hinweg sind die einzelnen Entwicklungsschritte erkennbar. Dadurch kam es zu einer Schwerpunktverlagerung mit angepasster Zielsetzung, da sich mit der Zeit herausstellte, welche Eigenschaften in früheren Versionen zwar Sinn ergaben, nun aber umgedacht werden mussten. Es kamen dabei eine Reihe von Managementmethoden zum Einsatz, die im Seminar Projektmanagement vorgestellt wurden.

Außerdem ist es uns gelungen, eine graphische Gestaltung für die Benutzeroberfläche zu erstellen, die modern und offen wirkt. Benutzerfreundlichkeit und Ästhetik sind ebenso vorhanden, genauso ist eine Reihe von Grundfunktionalitäten einfach zu erreichen und zu bedienen. Zudem bietet das Benachrichtigungsmanagementsystem der Patienten eine hervorragende Möglichkeit, als Facharzt sämtliche Kontakt Personen eines Patienten zu informieren, mit möglichst wenig Aufwand dabei.

Im Hinblick auf die Entwicklungsphase im Backend können wir auf eine klar strukturierte Architektur zurückblicken, in der bspw. der Aufbau der Patientendaten sowie der Hardware klar strukturiert ist. Dies bietet Raum für Erweiterbarkeit durch weitere Sensoren und Patienteninformationen. Die vorherige Analyse und Strukturierung durch UML-Diagramme ermöglichte es uns, eine einfache Art das Backend zu gestalten.

Besonders stolz sind wir auf die Erstellung einer ausführlichen Projektdokumentation. Diese dient als wertvoller Leitfaden und stellt sicher, dass alle Arbeitsschritte und Entscheidungen nachvollziehbar dokumentiert sind. Sie bildet die Basis für zukünftige Weiterentwicklungen und gewährleistet die Kontinuität des Projekts.

Insgesamt sind wir als Projektteam sehr zufrieden mit den erzielten Ergebnissen. Die erfolgreichen Umsetzungen in den genannten Bereichen legen ein starkes Fundament für die Weiterentwicklung der Smart Health Jewelry und deren möglichen Integration in den Gesundheitsmarkt.

## Zukünftige Optimierungen

Trotz der erfolgreichen Umsetzung unseres Projekts gibt es mehrere Bereiche, die wir beim nächsten Mal optimieren würden. Eine klarere Arbeitsaufteilung würde die Effizienz und Produktivität weiter steigern. Die anfänglichen Zielsetzungen haben sich im Laufe des Projekts verschoben, daher ist es wichtig, zukünftig flexibel auf Veränderungen zu reagieren und die Zielsetzungen regelmäßig zu überprüfen und anzupassen.

Ein weiterer wichtiger Punkt ist die bessere Datenaufbereitung, um fundiertere Rückschlüsse ziehen zu können. Es wurden zahlreiche Daten gesammelt, aufgrund mangelnder Zeit jedoch wenig Rückschlüsse gezogen. Ebenso nahmen im Verlauf die regelmäßigen Scrum-Meetings ab, was die Kommunikation und Koordination beeinträchtigten. Daher sollten diese Meetings konsequent und regelmäßig durchgeführt werden, um den Fortschritt kontinuierlich zu überwachen und zu steuern.

Wir standen vor der Herausforderung, dass die Hardware aufgrund von Kostengründen limitiert war. Eine frühere und gründlichere Evaluierung der Hardwareanforderungen könnte zukünftig helfen, diese Limitierungen zu minimieren. Ebenso stellte sich heraus, dass die erhaltenen Sensoren teilweise ungenau waren, was die Qualität der erfassten Daten beeinträchtigte. Hier ist es wichtig, beim nächsten Mal eine sorgfältigere Kaufentscheidung zu treffen, um sicherzustellen, dass die Sensoren den erforderlichen Standards entsprechen.

Des Weiteren würde eine bessere Trennung zwischen Frontend und Backend helfen, die Entwicklung und Wartung des Systems zu vereinfachen und die Verantwortlichkeiten im Team klarer definieren.

## Eigenständigkeitserklärung

Wir erkläre hiermit, dass wir beim Einsatz von IT-/KI-gestützten Schreibwerkzeugen diese Werkzeuge in der Rubrik „Übersicht verwendeter Hilfsmittel“ mit ihrem Produktnamen, meiner Bezugsquelle und einer Übersicht des im Rahmen dieser Studienarbeit genutzten Funktionsumfangs vollständig aufgeführt haben. Davon ausgenommen sind diejenigen IT-/KI gestützten Schreibwerkzeuge, die vom zuständigen Prüfer / von der zuständigen Prüferin zum Zeitpunkt der Abgabe meiner Studienleistung als nicht anzeigepflichtig eingestuft wurden („Whitelist“). Bei der Erstellung der vorgelegten Studienleistung haben wir durchgehend eigenständig und beim Einsatz IT-/KI-gestützter Schreibwerkzeuge steuernd gearbeitet.

---

Eduard Wegele

---

Dennis Apelt

---

Maxime Riebschläger

---

Linus Brüstle