

**Inhaltsverzeichnis**

[Abbildungsverzeichnis 1](#_Toc220669057)

[Tabellenverzeichnis 2](#_Toc220669058)

[Listingverzeichnis 3](#_Toc220669059)

[Abkürzungsverzeichnis 4](#_Toc220669060)

[1. Einleitung 5](#_Toc220669061)

[1.1 Projektbeschreibung 5](#_Toc220669062)

[1.2 Projektumfeld 5](#_Toc220669063)

[1.3 Projektziel 6](#_Toc220669064)

[1.4 Projektbegründung 7](#_Toc220669065)

[1.5 Projektschnittstellen 7](#_Toc220669066)

[1.5.1 Technische Schnittstellen 7](#_Toc220669067)

[1.5.2 Organisatorische Schnittstellen 7](#_Toc220669068)

[1.6 Projektabgrenzung 8](#_Toc220669069)

[1.7 Abweichungen vom Projektauftrag 8](#_Toc220669070)

[2. Projektplanung 9](#_Toc220669071)

[2.1 Vorgehensmodell 9](#_Toc220669072)

[2.2 Ressourcen- und Ablaufplanung 9](#_Toc220669073)

[2.2.1 Zeitplanung 9](#_Toc220669074)

[2.2.2 Kostenplanung 9](#_Toc220669075)

[2.3 Risikoanalyse 10](#_Toc220669076)

[3. Technische Umsetzung 11](#_Toc220669077)

[3.1 Architektur und Design 11](#_Toc220669078)

[3.1.1 Architektur-Konzept 11](#_Toc220669079)

[3.1.2 Datenfluss 11](#_Toc220669080)

[3.2 Auswahl der Technologien 12](#_Toc220669081)

[3.2.1 Software und Hardware 12](#_Toc220669082)

[3.3 Implementierung 13](#_Toc220669083)

[3.3.1 Kernkomponenten der Simulation 13](#_Toc220669084)

[3.3.2 Zufallselemente und Verteilung 14](#_Toc220669085)

[3.4 Benutzeroberfläche 15](#_Toc220669086)

[3.4.1 UI-Konzept und Usability 15](#_Toc220669087)

[3.4.2 Elemente und Animation 15](#_Toc220669088)

[4. Qualitätssicherung 17](#_Toc220669089)

[4.1 Testkonzept 17](#_Toc220669090)

[4.1.1 Teststrategie 17](#_Toc220669091)

[4.1.2 Testumgebung 17](#_Toc220669092)

[4.2 Testdurchführung 17](#_Toc220669093)

[4.2.1 Unit-Tests 17](#_Toc220669094)

[4.2.2 Integrationstests 18](#_Toc220669095)

[4.2.3 Blackbox-Tests (nach ISO 29119) 18](#_Toc220669096)

[4.3 Fehlerbehandlung 19](#_Toc220669097)

[4.3.1 Eingabevalidierung 19](#_Toc220669098)

[4.3.2 Exception-Handling 19](#_Toc220669099)

[4.4 Qualitätskriterien 19](#_Toc220669100)

[4.4.1 Code-Qualität 19](#_Toc220669101)

[4.4.2 Dokumentationsgrad 20](#_Toc220669102)

[4.5 Testergebnisse 20](#_Toc220669103)

[5. Fazit und Ausblick 21](#_Toc220669104)

[5.1 Projektzusammenfassung 21](#_Toc220669105)

[5.1.1 Zielerreichung 21](#_Toc220669106)

[5.1.2 Projektverlauf 22](#_Toc220669107)

[5.1.3 Herausforderungen und Lösungen 22](#_Toc220669108)

[5.2 Persönliche Reflexion 22](#_Toc220669109)

[5.2.1 Erkenntnisse und Lernfolge 22](#_Toc220669110)

[5.2.2 Bewertung der Architektur 23](#_Toc220669111)

[5.3 Ausblick 23](#_Toc220669112)

[Literaturverzeichnis 24](#_Toc220669113)

[Anhang I](#_Toc220669114)

[A1 Projektplanung I](#_Toc220669115)

[A1.1 Zeitplanung I](#_Toc220669116)

[A1.2 Arbeitspaketliste II](#_Toc220669117)

[A2 Kosten & Ressourcen IV](#_Toc220669118)

[A2.1 Kostenaufstellung nach Projektphasen IV](#_Toc220669119)

[A2.2 Detaillierte Kosten nach Arbeitspaketen IV](#_Toc220669120)

[A2.3 Ressourcen-Kostenübersicht VI](#_Toc220669121)

[A3 Mockups VII](#_Toc220669122)

[A3.1 Hauptseite VII](#_Toc220669123)

[A3.2 Statistikseite VIII](#_Toc220669124)

[A3.3 Parlamentsseite IX](#_Toc220669125)

[A4 Nutzwertanalysen X](#_Toc220669126)

[A4.1 IDE X](#_Toc220669127)

[A4.2 Build-Tools X](#_Toc220669128)

[A4.3 GUI-Framework XI](#_Toc220669129)

[A5 Technische Details XII](#_Toc220669130)

[A5.1 Use-Case-Diagramm XII](#_Toc220669131)

[A5.2 Paketdiagramm XIII](#_Toc220669132)

[A5.3 Klassendiagramm Model-Layer XIV](#_Toc220669133)

[A5.4 Klassendiagramm Kalkulationslogik XIV](#_Toc220669134)

[A5.5 Klassendiagramm Core XV](#_Toc220669135)

[A5.6 Klassendiagramm Skandale XV](#_Toc220669136)

[A5.7 Klassendiagramm View Komponenten XVI](#_Toc220669137)

[A6 Unit-Tests XVII](#_Toc220669138)

[A7 Testprotokoll XX](#_Toc220669139)

[A8 Finale Anwendung XXI](#_Toc220669140)

[A8.1 Start-Screen XXI](#_Toc220669141)

[A8.2 Main-Screen XXII](#_Toc220669142)

[A8.3 Statistik-Screen XXIII](#_Toc220669143)

[A8.4 Parlament-Screen XXIV](#_Toc220669144)

[A9 Listings XXV](#_Toc220669145)

[A9.1 Kernschleife der SimulationEngine XXV](#_Toc220669146)

[A9.2 Entscheidungslogik in VoterBehavior XXVI](#_Toc220669147)

[A9.3 Generierung von Skandal-Events XXVII](#_Toc220669148)

[A9.4 Abklingalgorithmus der Exponentialverteilung XXVII](#_Toc220669149)

[A9.5 Eingabevalidierung XXVIII](#_Toc220669150)

[A9.6 Daten-Update der Diagramme XXIX](#_Toc220669151)

[A9.7 Partikel-Rendering-Loop XXX](#_Toc220669152)

# Abbildungsverzeichnis

[Abbildung 1: Gantt-Diagramm I](#_Toc220669153)

[Abbildung 2: Mockup Hauptseite VII](#_Toc220669154)

[Abbildung 3: Mockup Statistikseite VIII](#_Toc220669155)

[Abbildung 4: Mockup Parlamentsseite IX](#_Toc220669156)

[Abbildung 5: Use-Case-Diagramm XII](#_Toc220669157)

[Abbildung 6: Paketdiagramm XIII](#_Toc220669158)

[Abbildung 7: Klassendiagramm Model-Layer XIV](#_Toc220669159)

[Abbildung 8: Klassendiagramm Kalkulation XIV](#_Toc220669160)

[Abbildung 9: Klassendiagramm Core XV](#_Toc220669161)

[Abbildung 10: Klassendiagramm Skandal XV](#_Toc220669162)

[Abbildung 11:Klassendiagramm View Komponenten XVI](#_Toc220669163)

[Abbildung 12: Start-Screen XXI](#_Toc220669164)

[Abbildung 13: Main-Screen XXII](#_Toc220669165)

[Abbildung 14: Statistik-Screen XXIII](#_Toc220669166)

[Abbildung 15: Parlament-Screen XXIV](#_Toc220669167)

# Tabellenverzeichnis

[Tabelle 1: Zeitliche Verteilung der Projektphasen 9](#_Toc220669168)

[Tabelle 2: Risikoauflistung 10](#_Toc220669169)

[Tabelle 3: Testumgebung: 17](#_Toc220669170)

[Tabelle 4: Eingabevalidierung 19](#_Toc220669171)

[Tabelle 5: Testergebnisse-Übersicht 20](#_Toc220669172)

[Tabelle 6: Anforderungserfüllung 21](#_Toc220669173)

[Tabelle 7: Soll-Ist-Vergleich Zeitplanung 22](#_Toc220669174)

[Tabelle 8: Arbeitspaketlist III](#_Toc220669175)

[Tabelle 9:Kostenaufstellung nach Projektphasen IV](#_Toc220669176)

[Tabelle 10: Detaillierte Kosten nach Arbeitspaketen V](#_Toc220669177)

[Tabelle 11: Ressourcen-Kostenübersicht VI](#_Toc220669178)

[Tabelle 12: Nutzwertanalyse IDE X](#_Toc220669179)

[Tabelle 13: Nutzwertanalyse Build-Tools X](#_Toc220669180)

[Tabelle 14: Nutzwertanalyse GUI-Framework XI](#_Toc220669181)

[Tabelle 15: Unit-Tests XIX](#_Toc220669182)

# Listingverzeichnis

[Listing 1: Kernschleife der SimulationEngine XXV](#_Toc220669183)

[Listing 2:Entscheidungslogik XXVI](#_Toc220669184)

[Listing 3: Generierung Skandal-Event XXVII](#_Toc220669185)

[Listing 4: Abklingalgorithmus XXVII](#_Toc220669186)

[Listing 5: Eingabevalidierung XXVIII](#_Toc220669187)

[Listing 6: Daten-Update der Diagramme XXIX](#_Toc220669188)

[Listing 7: Partikel-Rendering-Loop XXX](#_Toc220669189)

# Abkürzungsverzeichnis

CSS Cascading Style Sheet

GB Gigabyte

GUI Graphical User Interface

IDE Integrated Development Environment

IHK Industrie und Handelskammer

JRE Java Runtime Environment

JDK Java Development Kit

LTS Long Term Support

MVC Model-View-Controller

MVVM Model-View-View-Model

RAM Random Access Memory

UI User Interface

# Einleitung

Im Rahmen der Ausbildung zum Fachinformatiker für Anwendungsentwicklung erfolgt ein für die Berufsschule verpflichtendes Simulationsprojekt.

Es dient zur Überprüfung des Wissen- und Ausbildungsstands sowie zur Vorbereitung auf die Abschlussprüfung.

## Projektbeschreibung

Schulen vermitteln im Fachbereich Politik/Wirtschaft das Thema „Wahlsystem in Deutschland" als verpflichtenden Unterrichtsinhalt. Aktuell erfolgt die Vermittlung dieses Themas hauptsächlich durch statische Medien wie Lehrbücher, Arbeitsblätter und Lehrvideos. Eine typische Unterrichtsstunde gliedert sich dabei wie folgt:

Der Lehrer erklärt zunächst Wahlverhalten anhand statischer Diagramme, anschließend analysieren die Schüler historische Wahlergebnisse in Tabellenform und abschließend erfolgt eine Diskussion über Einflussfaktoren anhand theoretischer Beispiele.

Aufgrund dieser Lehrweise haben viele Lernende Probleme, ein tieferes Verständnis für das Thema zu entwickeln. Die fehlende Interaktivität verhindert das Experimentieren mit verschiedenen Szenarien, wodurch Zusammenhänge schwer nachvollziehbar bleiben. Die statischen Medien zeigen nicht, wie sich dynamische Faktoren wie Skandale oder Kampagnenbudgets konkret auswirken.

Aus diesem Grund soll eine interaktive Visualisierung entwickelt werden, welche die Lernenden durch explorative Interaktion anspricht. Die Anwendung soll das Konzept der Wählerwanderung durch Echtzeit-Simulationen erlebbar machen und damit die didaktische Qualität des Unterrichts steigern.

## Projektumfeld

Die Projektdurchführung erfolgt als schulische Projektarbeit direkt beim Auftraggeber, der „Beruflichen Schule der Hanse- und Universitätsstadt Rostock -Technik-". Die Anwendung richtet sich an Berufsschulklassen und der Einsatz erfolgt in den Computerräumen der Schule.

Die Zielplattform der Anwendung bilden die schuleigenen Windows Clients. Eine besondere technische Rahmenbedingung stellen die fehlenden Admin-Rechte dar, was die Installation von JRE durch Schüler oder Lehrer ausschließt. Diese Einschränkung erfordert eine Standalone-Lösung mit eingebettetem JRE.

## Projektziel

Ziel ist es, eine Desktop-Anwendung zu entwickeln, mit der Schüler Wählerwanderungen und externe Einflüsse bei Wahlen interaktiv simulieren können. Die Anwendung ergänzt den bestehenden Politikunterricht durch praktische Experimente und visualisiert Prozesse, die in Lehrbüchern nur linear dargestellt werden.

Die Anwendung muss folgende Kernfunktionalitäten bereitstellen:

**Konfiguration und Simulation:**

* Eingabe von Simulationsparametern
* Simulation von Wählerwanderungen auf Basis von Normalverteilung, Gleichverteilung und Exponentialverteilung
* Zufallsereignisse

**Visualisierung:**

* Echtzeit-Darstellung der Wahlergebnisse in Form von Balken- und Liniendiagrammen
* Animation der Wählerbewegungen zwischen Parteien als farbige Partikel
* Ereignis-Feed, der alle relevanten Vorkommnisse während der Simulation protokolliert

**Technische Anforderungen:**

* Auslieferung als eigenständige .exe mit eingebettetem JRE, um administrative Rechte auf den Schulrechnern zu umgehen
* Lauffähigkeit auf den vorhandenen Windows-11-PCs ohne zusätzliche Installation
* Intuitive Bedienung gemäß ISO 9241-110

Das Projekt gilt als erfolgreich abgeschlossen, wenn die Anwendung auf den Schulrechnern ohne Installation ausführbar ist, alle drei Zufallsverteilungen korrekt implementiert sind und die Echtzeit- Visualisierung flüssig läuft.

## Projektbegründung

Die Hauptschwachstelle bisheriger Lernmaterialien zu Wahlsystemen ist deren statischer Charakter. Lehrbücher, Grafiken und Videos zeigen zwar die Theorie, aber nicht die dynamischen Zusammenhänge einer Wahl. Durch das Projekt sollen nicht sichtbare Prozesse wie die Wählerwanderung oder externe Einflüsse visualisiert werden. Lernende können dadurch eigene Szenarien durchspielen und verstehen, wie sich Skandale oder Kampagnenbudgets auf das Wahlergebnis auswirken.

Ein weiteres Problem ist das fehlende direkte Feedback im traditionellen Unterricht. Die didaktische Qualität soll damit gesteigert werden, indem die Lernenden beispielsweise bei einer Parameteränderung direktes Feedback erhalten. Dieser iterative Lernprozess entspricht modernen didaktischen Prinzipien und ermöglicht es, Zusammenhänge durch aktives Experimentieren zu erfassen.

Hinzu kommen technische Einschränkungen der Schulinfrastruktur. Auf den Windows-11-PCs der Beruflichen Schule Rostock ist die Installation von Software, welche Adminrechte benötigen nicht möglich. Die Auslieferung als .exe mit eingebettetem JRE gewährleistet, dass die Anwendung ohne Installation lauffähig ist und auf allen Schulrechnern funktioniert.

Zudem steigert eine animierte Visualisierung von Wählerbewegungen die Motivation erheblich, da der Prozess der Meinungsbildung greifbar wird. Lehrkräfte profitieren durch die Zeitersparnis, da aufwändige Tafelzeichnungen oder Excel-Tabellen entfallen.

## Projektschnittstellen

### Technische Schnittstellen

Da die Software als Standalone-Anwendung konzipiert ist, existieren keine Schnittstellen zu vorhandenen IT-Infrastrukturen oder Datenbanken.

### Organisatorische Schnittstellen

Der fachliche Ansprechpartner des Kunden ist Herr Patett als Hauptverantwortlicher.

## Projektabgrenzung

Durch den begrenzten Zeitumfang des Projektes von 120 Stunden wurde der Fokus auf die Kernfunktionalität gelegt.

Eine Bereitstellung für weitere Betriebssysteme erfolgt nicht, da die Schulrechner ausschließlich Windows 11 nutzen und eine plattformübergreifende Entwicklung den Projektumfang überschreiten würde.

Es werden keine Simulationsdaten in einer Datenbank gespeichert, um die Architektur zu vereinfachen und Datenschutzprobleme bei der Nutzung auf Schulrechnern zu vermeiden.

Eine Implementierung von KI-Algorithmen zur Vorhersage von Wählerverhalten erfolgt nicht, da mathematische Zufallsverteilungen für didaktische Zwecke ausreichend realistische Ergebnisse liefern.

Zudem werden keine realen historischen Wahldaten verwendet, um didaktische Neutralität zu gewährleisten und urheberrechtliche Probleme zu vermeiden.

## Abweichungen vom Projektauftrag

Im Verlauf des Projekts traten keine inhaltlichen Abweichungen vom genehmigten Projektauftrag auf. Die ursprünglich geplanten 120 Arbeitsstunden wurden exakt eingehalten, wobei sich der tatsächliche Aufwand in den einzelnen Phasen minimal verschob, siehe Tabelle 7: Soll-Ist-Vergleich Zeitplanung

# Projektplanung

## Vorgehensmodell

Für die Durchführung dieses Projekts wurde das Wasserfallmodell als lineares, sequenzielles Vorgehensmodell gewählt. Die Wahl des Wasserfallmodells begründet sich durch den festen Abgabetermin (05.02.2026) und die klar definierten Anforderungen aus dem Projektauftrag. Das Wasserfallmodell sieht vor, dass die Entwicklung in aufeinanderfolgenden, klar abgegrenzten Phasen erfolgt. Dieser Ansatz ermöglicht eine präzise Planung und Kontrolle des Projektfortschritts sowie eine umfassende Dokumentation jeder Entwicklungsphase entsprechend der IHK-Anforderungen.

## Ressourcen- und Ablaufplanung

### Zeitplanung

Die Projektdurchführung erstreckt sich über einen Zeitraum von 21,6 Wochen (06.09.2025 bis 05.02.2026) und ist in vier Hauptphasen unterteilt:

|  |  |  |
| --- | --- | --- |
| **Phasen** | **Beschreibung** | **Zeitraum** |
| Phase 1 | Projektplanung | Sep. 2025 |
| Phase 2 | Technische Umsetzung | Okt. – Nov. 2025 |
| Phase 3 | Qualitätssicherung | Dez. 2025 – Jan. 2026 |
| Phase 4 | Projektabschluss | Jan. – Feb. 2026 |

Tabelle : Zeitliche Verteilung der Projektphasen

Eine detaillierte Visualisierung der Zeitplanung mit allen Arbeitspaketen, Abhängigkeiten und Meilensteinen ist im Anhang A1.1 Zeitplanung (Abbildung 1: Gantt-Diagramm) als Gantt-Diagramm dargestellt.

### Kostenplanung

Die Kostenplanung basiert auf einer Kalkulation der Personalkosten sowie der benötigten Sachmittel. Für die Projektdurchführung wird von einem Gesamtaufwand von 120 Arbeitsstunden ausgegangen. Bei einem Stundensatz von 6,50€ ergeben sich Personalkosten von 780,00€. Alle benötigten Software-Werkzeuge sind als Open-Source-Lösungen oder über Bildungslizenzen kostenfrei verfügbar, die erforderliche Hardware ist bereits vorhanden. Eine detaillierte Aufschlüsselung der Kosten nach Projektphasen befindet sich im Anhang A2(Tabelle 9,Tabelle 10,Tabelle 11)

## Risikoanalyse

Die folgende Tabelle zeigt mögliche Risiken während der Projektdurchführung und entsprechende Gegenmaßnahmen:

|  |  |  |  |
| --- | --- | --- | --- |
| **Risiko** | **Wahrscheinlichkeit** | **Auswirkung** | **Gegenmaßnahmen** |
| Performance-Problem bei hoher Wähleranzahl | Mittel | Hoch | Frühzeitige Belastungstests, Optimierung der Datenstruktur |
| Kompatibilitätsprobleme auf Schulrechnern | Gering | Hoch | Test auf Schulrechner, jpackage mit eingebettetem JRE |
| Fehlerhafte Zufallsverteilungen | Mittel | Mittel | Statistische Validierung, Unit-Tests |
| Zeitverzug durch unterschätze Komplexität | Mittel | Hoch | Pufferzeiten eingeplant, wöchentliche Fortschrittskontrolle |
| Nichterfüllung der IHK-Anforderungen | Gering | Hoch | Checkliste erstellen, regelmäßiger Abgleich mit Projektauftrag |

Tabelle : Risikoauflistung

# Technische Umsetzung

## Architektur und Design

### Architektur-Konzept

Die Umsetzung des Projekts erfolgt auf Basis des MVC- Architekturmusters. Dieses sieht eine Trennung der Anwendung in das Datenmodell, die Darstellung der Daten und die Steuerung des Programms vor.

Diese Teilung erfolgt, um die spätere Wartung und Erweiterung der Software zu vereinfachen. Die einzelnen Komponenten können mit geringem Aufwand angepasst oder ausgetauscht werden, ohne dass die anderen Bestandteile davon betroffen sind. Des Weiteren wird die Übersichtlichkeit und Testbarkeit des Quellcodes durch die Nutzung des MVC-Musters verbessert

Die Rolle der View wird von JavaFX FXML-Dateien eingenommen, welche die Benutzeroberfläche beschreiben. Der Controller wird durch Java Controller-Klassen realisiert, die zwischen View und Model vermitteln. Das Model der Anwendung wird von der JSON-basierten Datenspeicherung sowie den dazugehörigen Entity-Klassen abgebildet.

Als Alternative wurde das MVVM-Muster betrachtet, welches häufig bei JavaFX-Anwendungen zum Einsatz kommt. Jedoch wurde sich für MVC entschieden, da dieses Muster aufgrund der geringeren Komplexität besser zum Umfang und den Anforderungen des Projekts passt. Die direkte Kopplung zwischen View und Controller ist für die Verwaltung von Schülerdaten ausreichend und erleichtert die Entwicklung im vorgegebenen Zeitrahmen.

### Datenfluss

Im Folgenden wird der Datenfluss innerhalb der Anwendung beschrieben.

**Benutzereingaben und Parametersteuerung (View → Controller):**

Interaktionen des Benutzers, wie das Verschieben der Schieberegler für Medieneinfluss oder die Anpassung von Parteibudgets im Dashboard, werden vom DashboardController entgegengenommen. Diese Eingaben werden über den ParameterValidator auf ihre zulässigen Wertebereiche geprüft und anschließend direkt in die SimulationParameters des Models übertragen.

**Simulationszyklus und Logikverarbeitung (Controller → Model):**

SimulationEngine, welche den zentralen Simulations-Loop steuert. Innerhalb eines Zeitschritts findet ein kaskadierender Datenfluss im Model statt:

1. Der ScandalScheduler berechnet die Wahrscheinlichkeit für Ereignisse und modifiziert die scandalPenalty der betroffenen Party-Objekte.
2. VoterBehavior nutzt den VoterDecisionContext, um unter Berücksichtigung von VoterType-spezifischen Modifikatoren die neue Parteipräferenz für die VoterPopulation zu berechnen.
3. Die Ergebnisse werden in einem SimulationState-Snapshot aggregiert.

**Echtzeit-Visualisierung und Feedback (Model → Controller → View):**

Nach Abschluss der Berechnungen stellt das Model die aktualisierten Daten dem Controller zur Verfügung. Dieser verteilt die Informationen an die spezialisierten View-Komponenten:

* Der ChartManager aktualisiert die Balken- und Liniendiagramme für das direkte statistische Feedback.
* Der CanvasRenderer erhält die Informationen über Wählerwanderungen und animiert die Partikelströme flüssig auf dem UI-Layer.
* Der FeedManager gibt die generierten Skandale im Ereignis-Ticker aus.

Durch diese Entkopplung wird sichergestellt, dass die rechenintensive Simulation der Wählerpopulation die Reaktionsfähigkeit der Benutzeroberfläche nicht beeinträchtigt.

## Auswahl der Technologien

### Software und Hardware

Die Auswahl der Entwicklungswerkzeuge erfolgte anhand von Nutzwertanalysen, um die beste Eignung für das Projekt sicherzustellen. Die detaillierten Analysen mit Gewichtungen und Berechnungen befinden sich im Anhang A4 Nutzwertanalysen. Entwicklungsumgebung und Tools Die Nutzwertanalysen ergaben folgende Entscheidungen:

• **IDE**: IntelliJ IDEA Ultimate (8,85 Pkt.) - exzellenter JavaFX-Support und

umfassende Code-Analyse-Tools (Tabelle 12: Nutzwertanalyse )

• **Build-Tool**: Apache Maven (8,75 Pkt.) - Standardisierung und einfache XML-Konfiguration (Tabelle 13: Nutzwertanalyse Build-Tools)

• **GUI-Framework**: JavaFX (8,95 Pkt.) – moderne UI-Elemente, CSS-Styling und Performance für Echtzeit-Visualisierungen (Tabelle 14: Nutzwertanalyse GUI-Framework)

**Weitere Technologie-Entscheidungen**

• **JDK**: OpenJDK 21 LTS, kostenfrei, Long-Term-Support

• **Versionsverwaltung**: Git (lokal) – Industriestandard

• **Zielsystem**: Windows 10/11 Desktop

• **Auslieferung**: jpackage-Tool – erstellt .exe mit eingebettetem JRE, läuft ohne Installation

## Implementierung

### Kernkomponenten der Simulation

Die technische Umsetzung folgt dem MVC-Muster, um die Simulationslogik strikt von der Visualisierung zu trennen. Die Architektur wird von drei zentralen Komponenten getragen:

**SimulationEngine:**

Die SimulationEngine steuert als zentrale Instanz den Simulationszyklus. In jedem Zeitschritt koordiniert sie die Berechnungsmodule für Medieneinflüsse, Parteibudgets und Zufallsereignisse (siehe Listing 1: Kernschleife der SimulationEngine). Um die Reaktionsfähigkeit der GUI zu erhalten, erfolgt die Logikberechnung entkoppelt vom visuellen Render-Takt der JavaFX-Komponenten.

**Voter-Logik:**

Die Anwendung simuliert bis zu 2.000.000 individuelle Agenten der Klasse Voter.

* **Entscheidungsmodell:** Die Klasse VoterBehavior berechnet basierend auf dem VoterDecisionContext und typspezifischen Parametern die Wechselwahrscheinlichkeiten (siehe Listing 2:Entscheidungslogik).
* **Archetypen:** Über das Enum VoterType werden sechs wissenschaftlich fundierte Wählerprofile mit individuellen Modifikatoren für Medien- und Positions-Sensitivität abgebildet.

**Parteien- und Ereignissystem:**

Die politischen Akteure werden in der PartyRegistry verwaltet. Jedes Party-Objekt hält dynamische Werte für Budget und Skandalschäden.

* **Skandal-Mechanik:** Der ScandalScheduler generiert ereignisbasierte ScandalEvents, deren Auswirkung durch den ScandalImpactCalculator ermittelt wird (siehe Listing 3: Generierung Skandal-Event).
* **Regeneration:** Negative Effekte klingen über einen implementierten Dämpfungsfaktor zeitabhängig ab, um realistische Regenerationsphasen politischer Reputation abzubilden.

### Zufallselemente und Verteilung

Die Realitätsnähe der Wahlsimulation wird durch die Kombination von drei mathematischen Verteilungsmodellen erreicht.

**Normalverteilung**

**Einsatz:** Modellierung individueller Meinungsänderungen innerhalb der Wählerpopulation. Da extreme Positionswechsel statistisch seltener auftreten als geringfügige Schwankungen, sorgt die Normalverteilung für ein stabiles Systemverhalten.

**Implementierung:** Die Berechnung erfolgt über die Methode nextGaussian(). Die statistische Korrektheit wurde durch Unit-Tests verifiziert (Tabelle 15: Unit-Tests).

**Gleichverteilung**

**Einsatz**: Initialisierung von Parteipositionen sowie die zufällige Auswahl der von Ereignissen betroffenen Akteure.

**Implementierung:** Verwendung von nextDouble() zur Erzeugung von Werten innerhalb definierter Intervalle, um eine faire statistische Gewichtung sicherzustellen.

**Exponentialverteilung**

* **Einsatz:** Simulation des zeitabhängigen Abklingverhaltens von Skandaleffekten.
* **Mechanik:** Die Wirkung eines Ereignisses ist unmittelbar nach dem Eintreten am höchsten und nimmt über die Zeit ab (sieheListing 4: Abklingalgorithmus). Dieser Ansatz verhindert, dass kurzfristige Skandale die langfristige Simulation unverhältnismäßig verzerren.

## Benutzeroberfläche

### UI-Konzept und Usability

Bei der Gestaltung der Benutzeroberfläche stand die intuitive Bedienbarkeit für den Einsatz im Unterricht im Vordergrund. Das Design folgt einem modernen Ansatz im Dark-Mode, um die Aufmerksamkeit der Lernenden auf die leuchtenden Simulationspartikel zu lenken. Die Umsetzung erfolgte strikt nach den Interaktionsprinzipien der ISO 9241-110:

 **Aufgabenangemessenheit:** Die Oberfläche ist auf die wesentlichen Funktionen reduziert. Es werden nur Parameter angezeigt, die einen direkten Einfluss auf die Simulation haben, um kognitive Überlastung zu vermeiden.

 **Selbstbeschreibungsfähigkeit:** Alle Steuerelemente sind eindeutig beschriftet. Zusätzlich unterstützen Tooltips den Benutzer.

 **Steuerbarkeit:** Der Benutzer behält die volle Kontrolle über den Simulationsfluss. Die Anwendung kann über dedizierte Steuerelemente jederzeit pausiert, fortgesetzt oder vollständig zurückgesetzt werden.

 **Fehlertoleranz:** Durch eine Echtzeit-Validierung (siehe Listing 5: Eingabevalidierung) werden ungültige Werte, wie negative Budgets, sofort visuell markiert und abgefangen.

Das Layout ist in drei funktionale Bereiche unterteilt. Im linken Bereich erfolgt die Konfiguration der Parameter, die zentrale Mitte dient der dynamischen Visualisierung der Wählerströme und auf der rechten Seite protokolliert ein Live-Ticker alle relevanten Ereignisse und Skandale.

### Elemente und Animation

Die Visualisierung der Wahldynamik erfolgt über zwei komplementäre Ebenen, die den abstrakten Simulationsprozess für den Anwender greifbar machen.

**Echtzeit-Diagramme:** Zur statistischen Auswertung der Ergebnisse kommen dynamische Balken- und Liniendiagramme zum Einsatz, die über den ChartManager gesteuert werden.

**Balkendiagramm:** Visualisiert die aktuelle Stimmenverteilung zwischen den Parteien in Echtzeit.

**Liniendiagramm:** Dokumentiert den historischen Verlauf der Wählerstimmen über die gesamte Simulationsdauer, um Trends und die Auswirkungen von Skandalen sichtbar zu machen.

**Implementierung:** Um eine flüssige Darstellung zu garantieren, werden die Diagramme über den SimulationController in festen Intervallen mit aggregierten Daten aus dem Model versorgt (siehe Listing 6: Daten-Update der Diagramme).

**Partikel-basierte Wählerwanderung:** Das zentrale visuelle Element ist die Animation der Wählerströme. Aufgrund der hohen Anzahl von bis zu 2.000.000 Agenten wurde hierfür eine spezialisierte Performance-Strategie gewählt:

**Technik:** Anstatt ressourcenintensive JavaFX-Nodes zu nutzen, zeichnet der CanvasRenderer die Wähler als einzelne Partikel direkt auf ein Canvas-Element.

**Stichproben-Verfahren:** Um die CPU-Last zu minimieren und die Übersichtlichkeit zu wahren, wird eine repräsentative Stichprobe von 1.000 Wählern animiert, während die zugrunde liegenden Diagramme stets die Gesamtdatenmenge widerspiegeln.

**Animation:** Die Bewegung der Partikel zwischen den Parteizentren erfolgt über einen AnimationTimer (siehe Listing 7: Partikel-Rendering-Loop).

# Qualitätssicherung

## Testkonzept

### Teststrategie

Die Qualitätssicherung erfolgt durch eine dreistufige Testbatterie, die verschiedene Testebenen kombiniert. Das Ziel ist es, Fehler frühzeitig zu erkennen und die Funktionalität der Anwendung aus unterschiedlichen Perspektiven sicherzustellen.

Die Teststrategie umfasst frei Ebenen:

* **Unit-Tests:** Testen einzelner Klassen und Methoden isoliert
* **Integrationstests:** Prüfen des Zusammenspiels zwischen Komponenten
* **Blackbox-Tests:** Validierung der Gesamtfunktionalität aus Benutzersicht

### Testumgebung

|  |  |  |
| --- | --- | --- |
| **Komponente** | **Entwicklungsumgebung** | **Zielumgebung** |
| Hardware | Intel i9-12400K, 32GB RAM | Intel i5-13400 |
| Betriebssystem | Windows 11 Pro | Windows 11 Pro Education |
| Java-Version | OpenJDK 25 LTS | OpenJDK 21 LTS |
| Test-Framework | JUnit 5.10 | - |

Tabelle : Testumgebung:

## Testdurchführung

### Unit-Tests

Unit-Tests prüfen einzelne Methoden isoliert. Insgesamt wurden 17 Unit-Tests geschrieben, die kritische Funktionen abdecken: Zufallsverteilungen (8 Tests), SimulationEngine (12 Tests), Voter-Klasse (7 Tests), Party-Klasse (5 Tests), Controller (6 Tests) und View-Logik (4 Tests). Alle Unit-Tests sind im A6 Unit-Tests (Tabelle 15) detailliert aufgelistet.

**Beispiel**: Test der Normalverteilung Ein kritischer Test validiert die Normalverteilung statistisch: 100.000 Zufallswerte werden generiert und Mittelwert sowie Varianz berechnet. Der Test prüft, ob Mittelwert nahe 0 (±0,01) und Varianz nahe 1 (±0,02) liegt. Ergebnis: Test erfolgreich mit Mittelwert 0,0042 und Varianz 0,998.

### Integrationstests

Integrationstests prüfen das Zusammenspiel zwischen Controller, Model und View. Es wurden 8 Integrationstests durchgeführt:

* IT-01: Parameter-Validierung
* IT-02: Fehlerbehandlung bei ungültigen Eingaben
* IT-03: MVC-Kommunikation
* IT-04: Echtzeit-Update der Diagramme
* IT-05: Animation synchron mit Simulationsschritten
* IT-06: Ereignis-Feed zeigt Skandale korrekt an
* IT-07: Pause/Resume-Funktionalität
* IT-08: Reset setzt alle Komponenten zurück

**Ergebnis:** Alle 8 Integrationstests bestanden

### Blackbox-Tests (nach ISO 29119)

Die Blackbox-Tests wurden gemäß ISO/IEC/IEEE 29119 durchgeführt und validieren die Software aus Benutzersicht ohne Kenntnis der internen Implementierung.

Die vollständige Dokumentation mit detaillierten Testschritten, erwarteten und tatsächlichen Ergebnissen, sowie Testabdeckungsmatrizen befindet sich im separaten Testprotokoll im Anhang A7 Testprotokoll

## Fehlerbehandlung

### Eingabevalidierung

Alle Benutzereingaben werden vom Controller validiert, bevor sie an das Model übergeben werden. Validierungsregeln umfassen:

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Regel** | **Fehlermeldung** |
| Wähleranzahl | Muss > 0 <= 2.000.000 sein | "Wähleranzahl muss zwischen 1 und 2.000.000 liegen |
| Parteianzahl | Muss zwischen 2 und 8 liegen | „Mindestens 2, maximal 8 Parteien erlaubt“ |
| Kampagnenbudget | Muss >=0 sein | „Budget kann nicht negativ sein“ |
| Medieneinfluss | Muss zwischen 0% und 100% liegen | "Medieneinfluss: 0-100%" |

Tabelle : Eingabevalidierung

### Exception-Handling

Die Anwendung nutzt ein strukturiertes Exception-Handling:

* **ValidationException:** Bei ungültigen Benutzereingaben
* **SimulationException:** Bei Fehlern während der Simulation
* **RuntimeException:** Für unerwartete Fehler

Alle Exceptions werden vom Controller abgefangen und dem Benutzer als verständliche Fehlermeldung angezeigt.

## Qualitätskriterien

### Code-Qualität

Die Code-Qualität wurde während der Entwicklung kontinuierlich mit analysiert. SonarQube ist ein statisches Code-Analyse-Tool, das potenzielle Fehler, Code-Smells und Sicherheitslücken in Echtzeit erkennt.

**Angewendete Qualitätsmaßnahmen:**

* **Statische Code-Analyse:** SonarQube wurde verwendet, um Code-Smells und Wartbarkeitsprobleme frühzeitig zu identifizieren
* **Clean Code Prinzipien:** Lesbare Methodennamen, Single Responsibility Principle, geringe Komplexität, KISS-Principle
* **Code-Reviews:** Regelmäßige Überprüfung kritischer Komponenten (SimulationEngine, VoterBehavior)
* **Refactoring:** Iterative Verbesserung der Code-Struktur basierend auf SonarQube-Empfehlungen

**Ergebnis:** Die finale Code-Basis weist eine hohe Wartbarkeit und geringe technische Schulden auf. Kritische Bugs und Sicherheitslücken wurden eliminiert.

### Dokumentationsgrad

* **Javadoc:** Alle öffentlichen Klassen und Methoden dokumentiert
* **Inline-Kommentare:** Komplexe Algorithmen kommentiert
* **README:** Installationsanleitung und nutzungsweise

## Testergebnisse

|  |  |  |  |
| --- | --- | --- | --- |
| **Test-Kategorie** | **Anzahl** | **Bestanden** | **Erfolgsquote** |
| Unit-Tests | 17 | 17 | 100% |
| Integrationstests | 8 | 8 | 100% |
| Blackbox-Tests (ISO 29119) | 7 | 7 | 100% |
| **Gesamt** | **32** | **32** | **100%** |

Tabelle : Testergebnisse-Übersicht

Alle definierten Qualitätsziele wurden erreicht: Funktionalität, Zuverlässigkeit, Benutzerfreundlichkeit und Wartbarkeit. Die Anwendung erfüllt alle funktionalen und qualitativen Anforderungen.

# Fazit und Ausblick

## Projektzusammenfassung

### Zielerreichung

Das Projekt „Entwicklung einer interaktiven Wahlsimulation mit JavaFX“ wurde erfolgreich abgeschlossen. Alle im Projektauftrag definierten Anforderungen wurden umgesetzt:

|  |  |  |
| --- | --- | --- |
| **Anforderung** | **Umsetzung** | **Status** |
| Wahlsimulation mit Zufallsverteilung | Normalverteilung, Gleichverteilung, Exponentialverteilung implementiert | ✅ |
| Interaktive GUI mit JavaFX | Konfiguration, Diagramme, Animation, Ereignis-Feed | ✅ |
| Echtzeit-Visualisierung | Balke- und Liniendiagramme aktualisieren live | ✅ |
| Animation der Wählerbewegung | Visuelle Darstellung als bewegte Punkte | ✅ |
| Auslieferung als .exe-Datei | Jpackage mit eingebettetem JRE | ✅ |
| Lauffähigkeit auf Schulrechnern | Tests auf Windows 11 erfolgreich | ✅ |

Tabelle : Anforderungserfüllung

### Projektverlauf

Das Projekt verlief weitgehend nach Plan. Die Zeitplanung mit 120 Arbeitsstunden wurde eingehalten:

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Geplant** | **Tatsächlich** | **Abweichung** |
| Phase 1: Projektplanung | 20h | 22h | +2h (UML-Diagramme komplexer) |
| Phase 2: Technische Umsetzung | 65h | 68h | +3h (Animationsoptimierung) |
| Phase 3: Qualitätssicherung | 20h | 17h | -3h (weniger Fehler) |
| Phase 4: Projektabschluss | 15h | 13h | -2h (Dokumentation parallel) |
| **Gesamt** | **120h** | **120h** | **±0h** |

Tabelle : Soll-Ist-Vergleich Zeitplanung

### Herausforderungen und Lösungen

Während der Projektdurchführung traten zwei wesentliche Herausforderungen auf:

**Animation bei hoher Wähleranzahl:** Gelöst wurde dies durch eine Stichprobe von 1.000 Wählern für die Animation, während die Diagramme weiterhin alle Daten zeigen. Dies ermöglicht eine flüssige Animation auch bei großen Datenmengen.

**Kompatibilität mit Schulrechnern Schulrechner:** haben keine Admin-Rechte für Java-Installation. Das jpackage-Tool erstellt eine .exe mit eingebettetem JRE, sodass keine Installation erforderlich ist und die Anwendung direkt nach Download läuft.

## Persönliche Reflexion

### Erkenntnisse und Lernfolge

Das Projekt hat wichtige Lernerfolge gebracht:

**Technische Kompetenz**

* Vertiefte Kenntnisse in JavaFX
* Erfahrungen mit MVC-Architektur in einem realen Projekt
* Umgang mit Build-Tools
* Verwendung funktionsspezifische Libraries

**Methodische Kompetenz**

* Strukturierte Projektplanung mit Gantt-Diagramm
* Systematisches Testen
* Nutzwertanalysen
* Risikomanagement

**Organisatorische Kompetenz**

* Festlegen von festen Arbeitszeiten
* Gesunde Prokrastination
* Feed-Back-Gespräche mit anderen Entwicklern

### Bewertung der Architektur

Die Wahl des MVC-Patterns hat sich bewährt. Die klare Trennung ermöglichte ein zeitgleiches Arbeiten an GUI und Simulationslogik. Die Entscheidung für JavaFX war richtig Swing hätte die Animation und CSS-Gestaltung deutlich erschwert. Maven als Build-Tool hat sich als übersichtlicher als Gradle herausgestellt.

## Ausblick

Für zukünftige Versionen sind folgende Erweiterungen denkbar:

**Kurzfristige Erweiterungen:**

* Export-Funktion für Simulationsergebnisse als CSV
* Screenshot-Funktion für Diagramme
* Zusätzliche Ereignisse
* Speichern/Laden von Ereignissen

**Langfristige Erweiterungen:**

* Verschiedene Wahlsysteme
* Geografische Verteilung mit Wahlkreisen
* KI-gesteuerte Kampagnenstrategien

# Literaturverzeichnis

Apache Software Foundation. (2024). Von Maven — Build Tool Documentation: https://maven.apache.org/ abgerufen

Box, G. E., & Muller, M. E. (1958). A note on the generation of random normal deviates. In *Annals of Mathematical Statistics* (pp. 610-611). Volume 29, Issue 2.

GitHub Inc. (2024). Von Version Control with Git: https://docs.github.com/ abgerufen

ISO. (2020). ISO 9241-110:2020. In *Ergonomics of human-system interaction — Part 110: Interaction principles.* International Organization for Standardization.

ISO/IEC/IEEE. (2021). ISO/IEC/IEEE 29119-3:2021. In *Software and systems engineering — Software testing — Part 3: Test documentation and test completion information.* International Organization for Standardization.

ISO/IEC/IEEE. (2021). ISO/IEC/IEEE 29119-4:2021. In *Software and systems engineering — Software testing — Part 4: Test techniques.* International Organization for Standardization.

JetBrains. (2024). Von IntelliJ IDEA Ultimate — IDE für Java-Entwicklung: https://www.jetbrains.com/idea/ abgerufen

OpenJDK Foundation. (2024). Von OpenJDK 21 LTS Documentation: https://openjdk.org/projects/jdk/21/ abgerufen

Oracle Corporation. (2024). Von JavaFX Documentation and API Reference: https://openjfx.io/ abgerufen

Oracle Corporation. (2024). Von The Java Tutorials: Learning the Java Language: https://docs.oracle.com/javase/tutorial/ abgerufen

# Anhang

## A1 Projektplanung

### A1.1 Zeitplanung

Ein Bild, das Text, Screenshot, Reihe, Diagramm enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Gantt-Diagramm

### A1.2 Arbeitspaketliste

|  |  |  |  |
| --- | --- | --- | --- |
| **AP-Nr.** | **Arbeitspaket** | **Dauer** | **Phase** |
| AP1.1 | Anforderungsanalyse | 5h | Phase 1 |
| AP1.2 | UML-Diagramme erstellen | 8h | Phase 1 |
| AP1.3 | Mockups erstellen | 4h | Phase 1 |
| AP1.4 | Risikoanalyse | 3h | Phase 1 |
| AP2.1 | Backend-Architektur aufsetzen | 8h | Phase 2 |
| AP2.2 | Zufallsverteilung implementieren | 12h | Phase 2 |
| AP 2.3 | SimulationsEngine entwickeln | 15h | Phase 2 |
| AP 2.4 | JavaFX GUI erstellen | 18h | Phasen 2 |
| AP2.5 | Controller integrieren | 8h | Phasen 2 |
| AP2.6 | Animation implementieren | 10h | Phasen 2 |
| AP3.1 | Testkonzept schreiben | 4h | Phase 3 |
| AP3.2 | Unit-Tests schreiben | 8h | Phase 3 |
| AP3.3 | Performance-Tests durchführen | 5h | Phase 3 |
| AP3.4 | Fehler beheben | 3h | Phase 3 |
| AP3.5 | Refactoring | 5h | Phase 3 |
| AP4.1 | Projektdokumentation verfassen | 13h | Phase 4 |
| AP 4.2 | Executable erstellen | 2h | Phase 4 |
| **Gesamtaufwand** | | **120h** |  |

Tabelle : Arbeitspaketlist

## A2 Kosten & Ressourcen

### A2.1 Kostenaufstellung nach Projektphasen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Projektphase** | **Geplante Zeit** | **Personalkosten (€6,50/h)** | **Sachmittel** | **Gesamtphase** |
| Phase 1: Projektplanung | 20h | €130,00 | €70,00 | €200,00 |
| Phase 2: Technische Umsetzung | 65h | €455,50 | €227,50 | €650,00 |
| Phase 3: Qualitätssicherung | 20h | €130,00 | €70,00 | €200,00 |
| Phase: Projektabschluss | 15h | €97,50 | €52,50 | €150,00 |
| **Gesamt** | **120h** | **€780,00** | **€420,00** | **€1200,00** |

Tabelle :Kostenaufstellung nach Projektphasen

### A2.2 Detaillierte Kosten nach Arbeitspaketen

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **AP-Nr.** | **Arbeitspaket** | **Dauer** | **Personal** | **Sachmittel** | **Gesamt** | **Phase** |
| AP1.1 | Anforderungsanalyse | 5h | €32,50 | €17,50 | €50,00 | Phase 1 |
| AP1.2 | UML-Diagramme erstellen | 8h | €52,00 | €28,00 | €80,00 | Phase 1 |
| AP1.3 | Mockups erstellen | 4h | €26,00 | €14,00 | €40,00 | Phase 1 |
| AP1.4 | Risikoanalyse | 3h | €19,50 | €10,50 | €30,00 | Phase 1 |
| AP2.1 | Backend-Architektur aufsetzen | 8h | €52,00 | €28,00 | €80,00 | Phase 2 |
| AP2.2 | Zufallsverteilung implementieren | 12h | €78,00 | €42,00 | €120,00 | Phase 2 |
| AP2.3 | SimulationsEngine entwickeln | 15h | €97,50 | €52,50 | €150,00 | Phase 2 |
| AP2.4 | JavaFX GUI erstellen | 18h | €117,00 | €63,00 | €180,00 | Phase 2 |
| AP2.5 | Controller implementieren | 8h | €52,00 | €28,00 | €80,00 | Phase 2 |
| AP2.6 | Animation implementieren | 10h | €65,00 | €35,00 | €100,00 | Phase 2 |
| AP3.1 | Testkonzept schreiben | 4h | €26,00 | €14,00 | €40,00 | Phase 3 |
| AP3.2 | Unit-Tests schreiben | 8h | €52,00 | €28,00 | €80,00 | Phase 3 |
| AP3.3 | Perfomance-Tests durchführen | 5h | €32,50 | €17,50 | €50,00 | Phase 3 |
| AP3.4 | Fehler beheben | 3h | €19,50 | €10,50 | €30,00 | Phase 3 |
| AP3.5 | Refactoring | 5h | €32,50 | €17,50 | €50,00 | Phase 3 |
| AP4.1 | Projektdokumentation verfassen | 13h | €84,50 | €45,50 | €130,00 | Phase 4 |
| AP4.2 | Executable erstellen | 2h | €13,00 | €7,00 | €20,00 | Phase 4 |
| **Summe** |  | **120h** | **€780,00** | **€420,00** | **€1.200,00** |  |

Tabelle : Detaillierte Kosten nach Arbeitspaketen

### A2.3 Ressourcen-Kostenübersicht

|  |  |  |  |
| --- | --- | --- | --- |
| **Ressource** | **Typ** | **Kosten** | **Bemerkung** |
| Personal | Auszubildender | €780,00 | 120h\*€6,50/h |
| Hardware | Schularbeitsplatz | Vorhanden |  |
| Software | OpenJDK 21, IntelliJ, Maven, Git, JUnit | Kostenlos | Open Source + Community |
| Gemeinkosten | Strom/Netzwerk | €420,00 | 120h\*€3,50/h |
| **Gesamt** |  | **€1.200,00** |  |

Tabelle : Ressourcen-Kostenübersicht

## A3 Mockups

### A3.1 Hauptseite

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 2: Mockup Hauptseite

### A3.2 Statistikseite

Ein Bild, das Screenshot, Multimedia-Software, Text, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 3: Mockup Statistikseite

### A3.3 Parlamentsseite

Ein Bild, das Text, Screenshot, Multimedia-Software, Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Mockup Parlamentsseite

## A4 Nutzwertanalysen

### A4.1 IDE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Kriterium** | **Gewichtung** | **IntelliJ Ultimate** | **Elipse** | **NetBeans** | **VS Code** |
| Einfachheit | 30% | 9 | 7 | 8 | 9 |
| Java-Unterstützung | 25% | 10 | 8 | 8 | 7 |
| Kosten | 20% | 8 | 10 | 10 | 10 |
| Lernkurve | 15% | 8 | 7 | 8 | 9 |
| Erweiterbarkeit | 10% | 9 | 9 | 8 | 9 |
| Gesamtnutzwert | 100% | 8,85 | 7,85 | 8,35 | 8,45 |

Tabelle : Nutzwertanalyse IDE

### A4.2 Build-Tools

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kriterium** | **Gewichtung** | **Apache Maven** | **Gradle** | **Ant** |
| Konfigurationseinfachheit | 30% | 9 | 8 | 6 |
| Abhängigkeitsmanagement | 25% | 10 | 9 | 5 |
| Performance | 20% | 8 | 10 | 7 |
| Community/Standard | 15% | 9 | 8 | 6 |
| Flexibilität | 10% | 7 | 10 | 9 |
| Gesamtnutzwert | 10% | 8,75 | 8,65 | 6,25 |

Tabelle : Nutzwertanalyse Build-Tools

### A4.3 GUI-Framework

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kriterium** | **Gewichtung** | **JavaFX** | **Swing** | **SWT** |
| Moderne UI | 30% | 10 | 6 | 7 |
| Performance | 25% | 9 | 7 | 8 |
| Einfachheit | 20% | 9 | 8 | 6 |
| Cross-Plattform | 15% | 9 | 9 | 7 |
| Community | 10% | 9 | 8 | 7 |
| Gesamtnutzwert | 100% | 9,25 | 7,25 | 7,25 |

Tabelle : Nutzwertanalyse GUI-Framework

## A5 Technische Details

### A5.1 Use-Case-Diagramm

Ein Bild, das Text, Diagramm, Screenshot, parallel enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Use-Case-Diagramm

### A5.2 Paketdiagramm

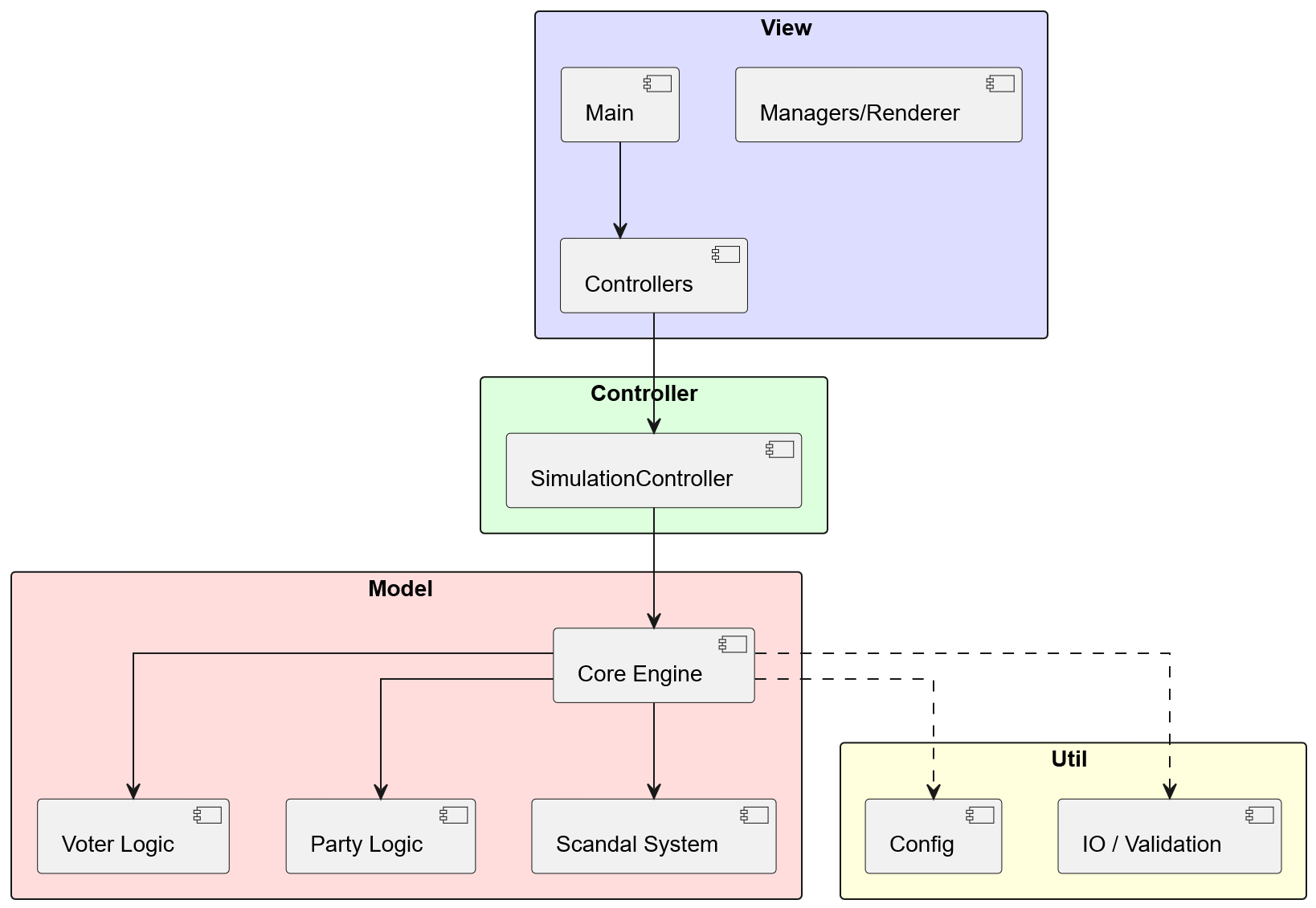


Abbildung : Paketdiagramm

### A5.3 Klassendiagramm Model-Layer

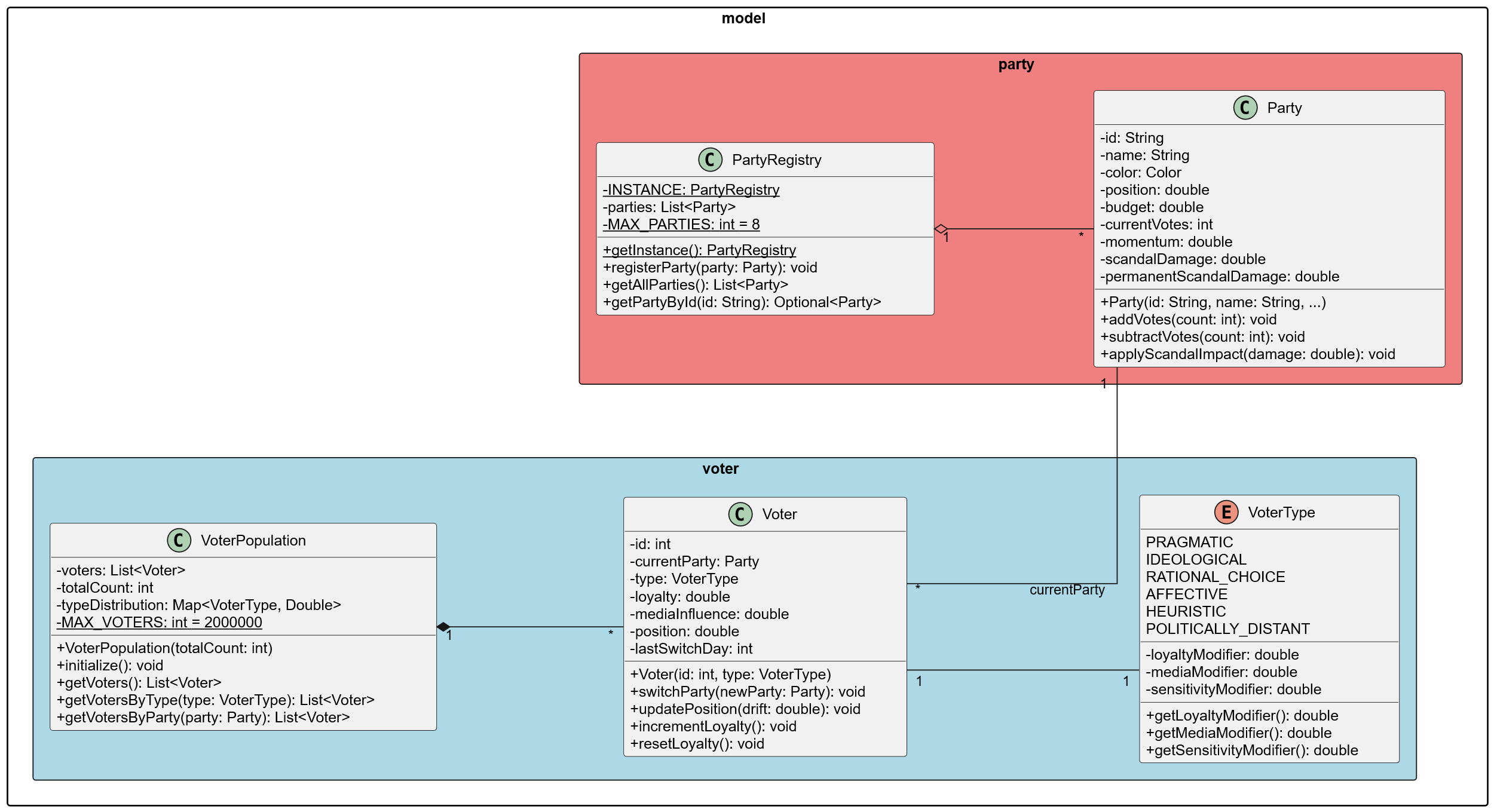


Abbildung : Klassendiagramm Model-Layer

### A5.4 Klassendiagramm Kalkulationslogik

Abbildung : Klassendiagramm Kalkulation

### A5.5 Klassendiagramm Core

Abbildung : Klassendiagramm Core

### A5.6 Klassendiagramm Skandale

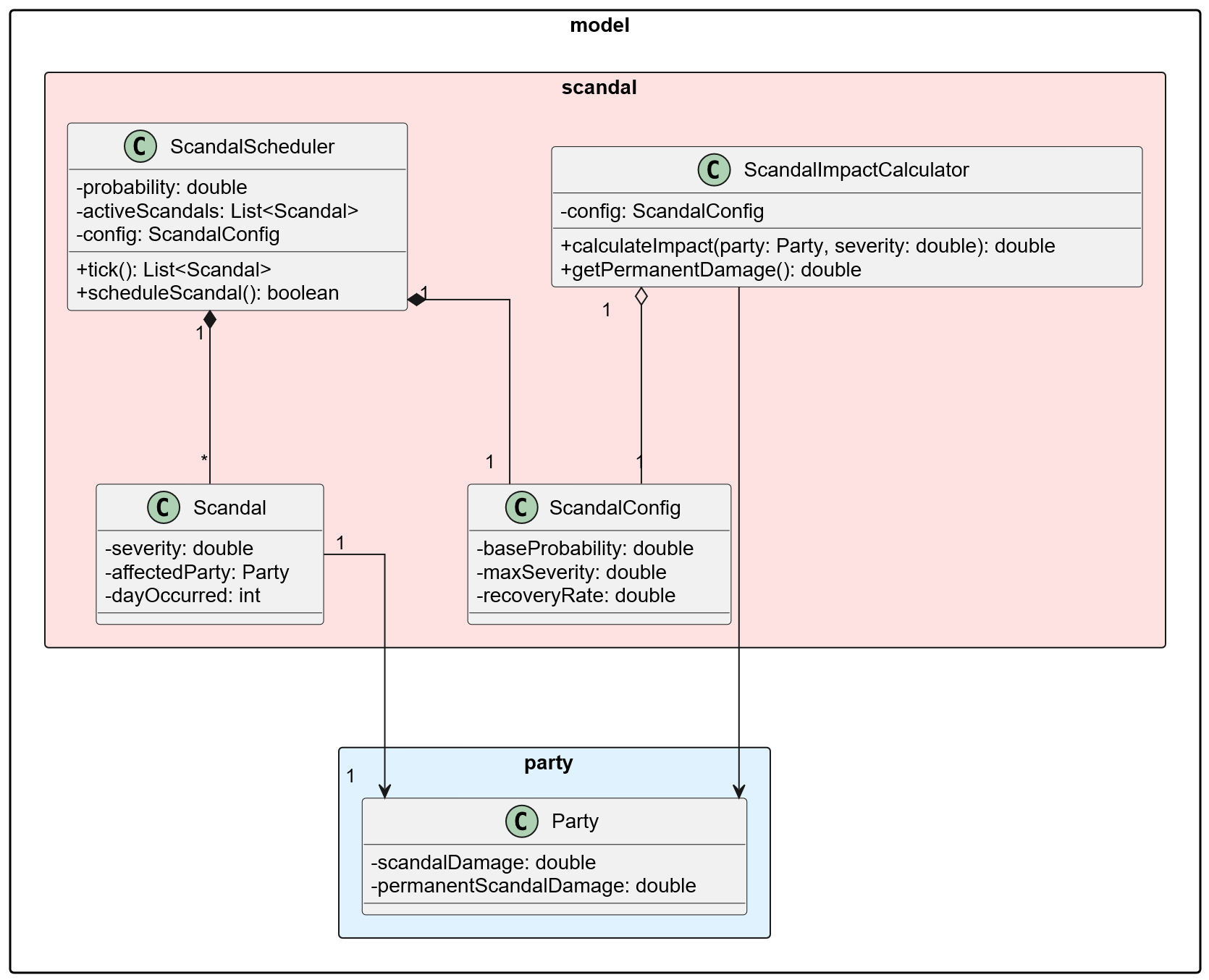


Abbildung : Klassendiagramm Skandal

## A5.7 Klassendiagramm View Komponenten

Abbildung :Klassendiagramm View Komponenten

## A6 Unit-Tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test-ID** | **Testklasse** | **Testmethode** | **Beschreibung** | **Status** |
| UT-01 | DistributionProviderTest | testNormalDistribution() | Prüft statistische Eigenschaften der Normalverteilung (μ=0, σ=1) | ✅ |
| UT-02 | DistributionProviderTest | testExponentialDistribution() | Validiert Exponentialverteilung mit λ=1.0 | ✅ |
| UT-03 | DistributionProviderTest | testUniformDistribution() | Testet Gleichverteilung im Intervall | ✅ |
| UT-04 | ParameterValidatorTest | testValidVoterCount() | Prüft gültige Wähleranzahl (1-2.000.000) | ✅ |
| UT-05 | ParameterValidatorTest | testInvalidVoterCountZero() | Validierung: 0 Wähler → Exception | ✅ |
| UT-06 | ParameterValidatorTest | testInvalidVoterCountNegative() | Validierung: Negative Wähler → Exception | ✅ |
| UT-07 | ParameterValidatorTest | testInvalidVoterCountTooHigh() | Validierung: > 2.000.000 Wähler → Exception | ✅ |
| UT-08 | ParameterValidatorTest | testValidPartyCount() | Prüft gültige Parteienanzahl (2-8) | ✅ |
| UT-09 | ParameterValidatorTest | testInvalidPartyCountTooFew() | Validierung: < 2 Parteien → Exception | ✅ |
| UT-10 | ParameterValidatorTest | testInvalidPartyCountTooMany | Validierung: > 8 Parteien → Exception | ✅ |
| UT-11 | ParameterValidatorTest | testValidBudget() | Prüft gültige Budgets (≥ 0) | ✅ |
| UT-12 | ParameterValidatorTest | testInvalidBudgetNegative() | Validierung: Negatives Budget → Exception | ✅ |
| UT-13 | VoterBehaviorTest | testOpinionChange() | Testet Meinungsänderung eines Wählers | ✅ |
| UT-14 | VoterBehaviorTest | testVoterTypeInfluence() | Validiert Einfluss des Wählertyps auf Entscheidung | ✅ |
| UT-15 | VoterBehaviorTest | testMediaInfluenceEffect() | Prüft Auswirkung von Medieneinfluss | ✅ |
| UT-16 | ScandalImpactCalculatorTest | testScandalImpactCalculation() | Berechnet Skandal-Auswirkungen auf Stimmen | ✅ |
| UT-17 | ScandalImpactCalculatorTest | testScandalDecayOverTime() | Testet exponentielles Abklingen eines Skandals | ✅ |

Tabelle : Unit-Tests

## A7 Testprotokoll

## A8 Finale Anwendung

### A8.1 Start-Screen

Ein Bild, das Text, Screenshot, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Start-Screen

### A8.2 Main-Screen

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Main-Screen

### A8.3 Statistik-Screen

Ein Bild, das Screenshot, Text, Farbigkeit, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Statistik-Screen

### A8.4 Parlament-Screen

Ein Bild, das Screenshot, Multimedia-Software, Software, Grafiksoftware enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung : Parlament-Screen

## A9 Listings

### A9.1 Kernschleife der SimulationEngine



Listing : Kernschleife der SimulationEngine

### A9.2 Entscheidungslogik in VoterBehavior



Listing :Entscheidungslogik

### A9.3 Generierung von Skandal-Events



Listing : Generierung Skandal-Event

### A9.4 Abklingalgorithmus der Exponentialverteilung



Listing : Abklingalgorithmus

### A9.5 Eingabevalidierung



Listing : Eingabevalidierung

### A9.6 Daten-Update der Diagramme



Listing : Daten-Update der Diagramme

### A9.7 Partikel-Rendering-Loop



Listing : Partikel-Rendering-Loop