

<b>IHK</b> Abschlussprüfung Teil 2 – Prüfungstermin	 Industrie- und Handelskammer zu Rostock
<b>IT- Berufe</b> <b>Deckblatt für die Dokumentation zur Schulischen Projektarbeit</b>	Ausbildungsberuf: <b>Fachinformatiker/-in Anwendungsentwicklung</b>

<b>Titel der Schulischen Projektarbeit:</b> Simulationsprojekt: „Wahlsimulation – Wie beeinflussen externe Faktoren das Ergebnis einer Wahl?“
--

<b>Prüfungsteilnehmer/in</b>	<b>Ausbildungs-/ Praktikumsunternehmen:</b>
<b>Vor- und Familienname:</b> <b>Nico Hoffmann</b>	<b>Verantwortliche/r für die Durchführung des betrieblichen Auftrages:</b> Klicken Sie hier, um Vor- und Zunamen des Ansprechpartners im Unternehmen einzugeben.
<b>Anschrift des Prüfungsteilnehmers:</b>	<b>Anschrift des Unternehmens:</b>
<b>OT:</b> Klicken Sie hier, um den Ortsteil einzugeben.	<b>OT:</b> Klicken Sie hier, um den Ortsteil einzugeben.
<b>Straße:</b> Papenhagen 18	<b>Straße:</b> Konrad-Zuse-Straße 1
<b>PLZ Ort</b> 18461, Richtenberg	<b>PLZ Ort</b> 18184, Roggentin
<b>Telefon:</b> +49 1758908108	<b>Telefon:</b> Klicken Sie hier, um die Telefonnr. des Ansprechpartners einzugeben.
<b>E-Mail:</b> dev.nhoffmann@proton.me	<b>E-Mail:</b> Klicken Sie hier, um die E-Mail-Adresse des Ansprechpartners einzugeben.

Ort \_\_\_\_\_ Klicken Sie hier, um ein Datum einzugeben.  
 Datum \_\_\_\_\_ Unterschrift Antragsteller/-in (Prüfling)

Ort \_\_\_\_\_ Klicken Sie hier, um ein Datum einzugeben.  
 Datum \_\_\_\_\_ Stempel/ Unterschrift Verantwortliche/r für den betrieblichen Auftrag



# Inhaltsverzeichnis

<b>Abbildungsverzeichnis.....</b>	<b>1</b>
<b>Tabellenverzeichnis .....</b>	<b>2</b>
<b>Listingverzeichnis.....</b>	<b>3</b>
<b>Abkürzungsverzeichnis .....</b>	<b>4</b>
<b>1. Einleitung.....</b>	<b>5</b>
1.1    Projektbeschreibung.....	5
1.2    Projektfeld.....	5
1.3    Projektziel .....	6
1.4    Projektbegründung .....	7
1.5    Projektschnittstellen .....	7
1.5.1        Technische Schnittstellen .....	7
1.5.2        Organisatorische Schnittstellen .....	7
1.6    Projektabgrenzung.....	8
1.7    Abweichungen vom Projektauftrag.....	8
<b>2. Projektplanung.....</b>	<b>9</b>
2.1    Vorgehensmodell.....	9
2.2    Ressourcen- und Ablaufplanung .....	9
2.2.1        Zeitplanung .....	9
2.2.2        Kostenplanung.....	9
2.3    Risikoanalyse .....	10
<b>3. Technische Umsetzung.....</b>	<b>11</b>
3.1    Architektur und Design.....	11
3.1.1        Architektur-Konzept.....	11
3.1.2        Datenfluss.....	11
3.2    Auswahl der Technologien.....	12
3.2.1        Software und Hardware.....	12
3.3    Implementierung .....	13
3.3.1        Kernkomponenten der Simulation .....	13
3.3.2        Zufallselemente und Verteilung .....	14
3.4    Benutzeroberfläche .....	15
3.4.1        UI-Konzept und Usability .....	15



3.4.2	Elemente und Animation .....	15
<b>4.</b>	<b>Qualitätssicherung .....</b>	<b>17</b>
4.1	Testkonzept .....	17
4.1.1	Teststrategie.....	17
4.1.2	Testumgebung .....	17
4.2	Testdurchführung .....	17
4.2.1	Unit-Tests .....	17
4.2.2	Integrationstests.....	18
4.2.3	Blackbox-Tests (nach ISO 29119).....	18
4.3	Fehlerbehandlung.....	19
4.3.1	Eingabevalidierung .....	19
4.3.2	Exception-Handling .....	19
4.4	Qualitätskriterien.....	19
4.4.1	Code-Qualität .....	19
4.4.2	Dokumentationsgrad .....	20
4.5	Testergebnisse .....	20
<b>5.</b>	<b>Fazit und Ausblick .....</b>	<b>21</b>
5.1	Projektzusammenfassung.....	21
5.1.1	Zielerreichung .....	21
5.1.2	Projektverlauf.....	22
5.1.3	Herausforderungen und Lösungen .....	22
5.2	Persönliche Reflexion .....	22
5.2.1	Erkenntnisse und Lernfolge .....	22
5.2.2	Bewertung der Architektur.....	23
5.3	Ausblick .....	23
<b>Literaturverzeichnis.....</b>	<b>24</b>	
<b>Anhang .....</b>	<b>I</b>	
A1	Projektplanung .....	I
A1.1	Zeitplanung .....	I
A1.2	Arbeitspaketliste .....	II
A2	Kosten & Ressourcen .....	IV
A2.1	Kostenaufstellung nach Projektphasen.....	IV



A2.2 Detaillierte Kosten nach Arbeitspaketen .....	IV
A2.3 Ressourcen-Kostenübersicht .....	VI
A3 Mockups .....	VII
A3.1 Hauptseite .....	VII
A3.2 Statistikseite .....	VIII
A3.3 Parlamentsseite .....	IX
A4 Nutzwertanalysen .....	X
A4.1 IDE .....	X
A4.2 Build-Tools .....	X
A4.3 GUI-Framework .....	XI
A5 Technische Details .....	XII
A5.1 Use-Case-Diagramm .....	XII
A5.2 Paketdiagramm .....	XIII
A5.3 Klassendiagramm Model-Layer .....	XIV
A5.4 Klassendiagramm Kalkulationslogik .....	XIV
A5.5 Klassendiagramm Core .....	XV
A5.6 Klassendiagramm Skandale .....	XV
A5.7 Klassendiagramm View Komponenten .....	XVI
A6 Unit-Tests .....	XVII
A7 Testprotokoll .....	XX
A8 Finale Anwendung .....	XXIII
A8.1 Start-Screen .....	XXIII
A8.2 Main-Screen .....	XXIV
A8.3 Statistik-Screen .....	XXV
A8.4 Parlament-Screen .....	XXVI
A9 Listings .....	XXVII
A9.1 Kernschleife der SimulationEngine .....	XXVII
A9.2 Entscheidungslogik in VoterBehavior .....	XXVIII
A9.3 Generierung von Skandal-Events .....	XXIX
A9.4 Abklingalgorithmus der Exponentialverteilung .....	XXIX
A9.5 Eingabevalidierung .....	XXX
A9.6 Daten-Update der Diagramme .....	XXXI



A9.7 Partikel-Rendering-Loop.....	XXXII
A10 Benutzerhandbuch.....	XXXIII



## Abbildungsverzeichnis

Abbildung 1: Gantt-Diagramm .....	I
Abbildung 2: Mockup Hauptseite .....	VII
Abbildung 3: Mockup Statistikseite.....	VIII
Abbildung 4: Mockup Parlamentsseite .....	IX
Abbildung 5: Use-Case-Diagramm.....	XII
Abbildung 6: Paketdiagramm .....	XIII
Abbildung 7: Klassendiagramm Model-Layer.....	XIV
Abbildung 8: Klassendiagramm Kalkulation.....	XIV
Abbildung 9: Klassendiagramm Core .....	XV
Abbildung 10: Klassendiagramm Skandal.....	XV
Abbildung 11:Klassendiagramm View Komponenten .....	XVI
Abbildung 12: Start-Screen.....	XXIII
Abbildung 13: Main-Screen.....	XXIV
Abbildung 14: Statistik-Screen .....	XXV
Abbildung 15: Parlament-Screen .....	XXVI



## Tabellenverzeichnis

Tabelle 1: Zeitliche Verteilung der Projektphasen .....	9
Tabelle 2: Risikoauflistung .....	10
Tabelle 3: Testumgebung: .....	17
Tabelle 4: Eingabevalidierung.....	19
Tabelle 5: Testergebnisse-Übersicht.....	20
Tabelle 6: Anforderungserfüllung .....	21
Tabelle 7: Soll-Ist-Vergleich Zeitplanung .....	22
Tabelle 8: Arbeitspaketlist.....	III
Tabelle 9:Kostenaufstellung nach Projektphasen.....	IV
Tabelle 10: Detaillierte Kosten nach Arbeitspaketen .....	V
Tabelle 11: Ressourcen-Kostenübersicht .....	VI
Tabelle 12: Nutzwertanalyse IDE .....	X
Tabelle 13: Nutzwertanalyse Build-Tools .....	X
Tabelle 14: Nutzwertanalyse GUI-Framework .....	XI
Tabelle 15: Unit-Tests .....	XIX



## **Listingverzeichnis**

Listing 1: Kernschleife der SimulationEngine .....	XXVII
Listing 2: Entscheidungslogik .....	XXVIII
Listing 3: Generierung Skandal-Event .....	XXIX
Listing 4: Abklingalgorithmus .....	XXIX
Listing 5: Eingabevalidierung .....	XXX
Listing 6: Daten-Update der Diagramme .....	XXXI
Listing 7: Partikel-Rendering-Loop .....	XXXII



## 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



## 1. 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.

### 1.1 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.

### 1.2 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.

### 1.3 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.



## 1.4 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.

## 1.5 Projektschnittstellen

### 1.5.1 Technische Schnittstellen

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

### 1.5.2 Organisatorische Schnittstellen

Der fachliche Ansprechpartner des Kunden ist Herr Patett als Hauptverantwortlicher.



## 1.6 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.

## 1.7 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



## 2. Projektplanung

### 2.1 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.

### 2.2 Ressourcen- und Ablaufplanung

#### 2.2.1 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	Projektabchluss	Jan. – Feb. 2026

Tabelle 1: 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.

#### 2.2.2 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)

## 2.3 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 unterschätzte Komplexität	Mittel	Hoch	Pufferzeiten eingeplant, wöchentliche Fortschrittskontrolle
Nichterfüllung der IHK-Anforderungen	Gering	Hoch	Checkliste erstellen, regelmäßiger Abgleich mit Projektauftrag

Tabelle 2: Risikoauflistung



## 3. Technische Umsetzung

### 3.1 Architektur und Design

#### 3.1.1 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.

#### 3.1.2 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.

## **3.2 Auswahl der Technologien**

### **3.2.1 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

## 3.3 Implementierung

### 3.3.1 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.

### 3.3.2 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 (siehe Listing 4: Abklingalgorithmus). Dieser



Ansatz verhindert, dass kurzfristige Skandale die langfristige Simulation unverhältnismäßig verzerren.

## 3.4 Benutzeroberfläche

### 3.4.1 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.

### 3.4.2 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).



## 4. Qualitätssicherung

### 4.1 Testkonzept

#### 4.1.1 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

#### 4.1.2 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 3: Testumgebung:

### 4.2 Testdurchführung

#### 4.2.1 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 ( $\pm 0,01$ ) und Varianz nahe 1 ( $\pm 0,02$ ) liegt. Ergebnis: Test erfolgreich mit Mittelwert 0,0042 und Varianz 0,998.

#### 4.2.2 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

#### 4.2.3 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



## 4.3 Fehlerbehandlung

### 4.3.1 Eingabevalidierung

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

Validierungsregeln

umfassen:

Parameter	Regel	Fehlermeldung
Wähleranzahl	Muss $> 0 \leq 2.000.000$ 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 $\geq 0$ sein	„Budget kann nicht negativ sein“
Medieneinfluss	Muss zwischen 0% und 100% liegen	"Medieneinfluss: 0-100%"

Tabelle 4: Eingabevalidierung

### 4.3.2 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.

## 4.4 Qualitätskriterien

### 4.4.1 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.

#### 4.4.2 Dokumentationsgrad

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

### 4.5 Testergebnisse

Test-Kategorie	Anzahl	Bestanden	Erfolgsquote
Unit-Tests	17	17	100%
Integrationstests	8	8	100%
Blackbox-Tests	7	7	100%
<b>Gesamt</b>	<b>32</b>	<b>32</b>	<b>100%</b>

Tabelle 5: Testergebnisse-Übersicht

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



## 5. Fazit und Ausblick

### 5.1 Projektzusammenfassung

#### 5.1.1 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 6: Anforderungserfüllung



### 5.1.2 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: Projektabchluss	15h	13h	-2h (Dokumentation parallel)
<b>Gesamt</b>	<b>120h</b>	<b>120h</b>	<b>±0h</b>

Tabelle 7: Soll-Ist-Vergleich Zeitplanung

### 5.1.3 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.

## 5.2 Persönliche Reflexion

### 5.2.1 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

#### **5.2.2 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.

### **5.3 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

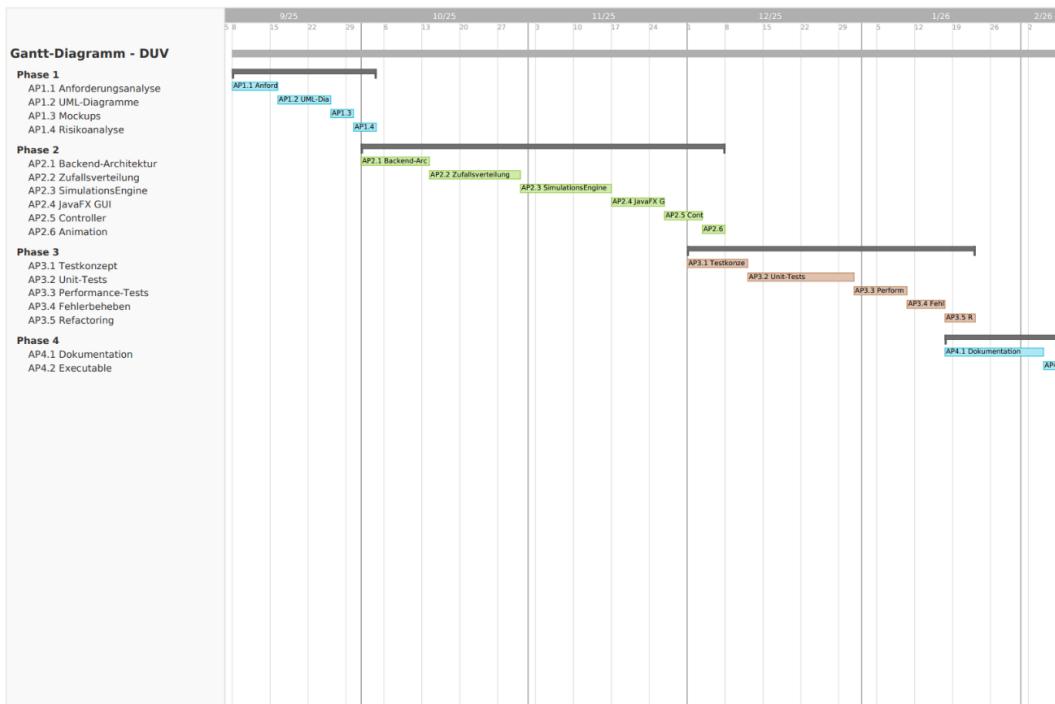


Abbildung 1: Gantt-Diagramm



## 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
AP2.3	SimulationsEngine entwickeln	15h	Phase 2
AP2.4	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
	<b>Gesamtaufwand</b>	<b>120h</b>	

Tabelle 8: 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: Projektabchluss	15h	€97,50	€52,50	€150,00
<b>Gesamt</b>	<b>120h</b>	<b>€780,00</b>	<b>€420,00</b>	<b>€1200,00</b>

Tabelle 9: 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
<b>Summe</b>		<b>120h</b>	<b>€780,00</b>	<b>€420,00</b>	<b>€1.200,00</b>	

Tabelle 10: 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
<b>Gesamt</b>		<b>€1.200,00</b>	

Tabelle 11: Ressourcen-Kostenübersicht



## A3 Mockups

### A3.1 Hauptseite

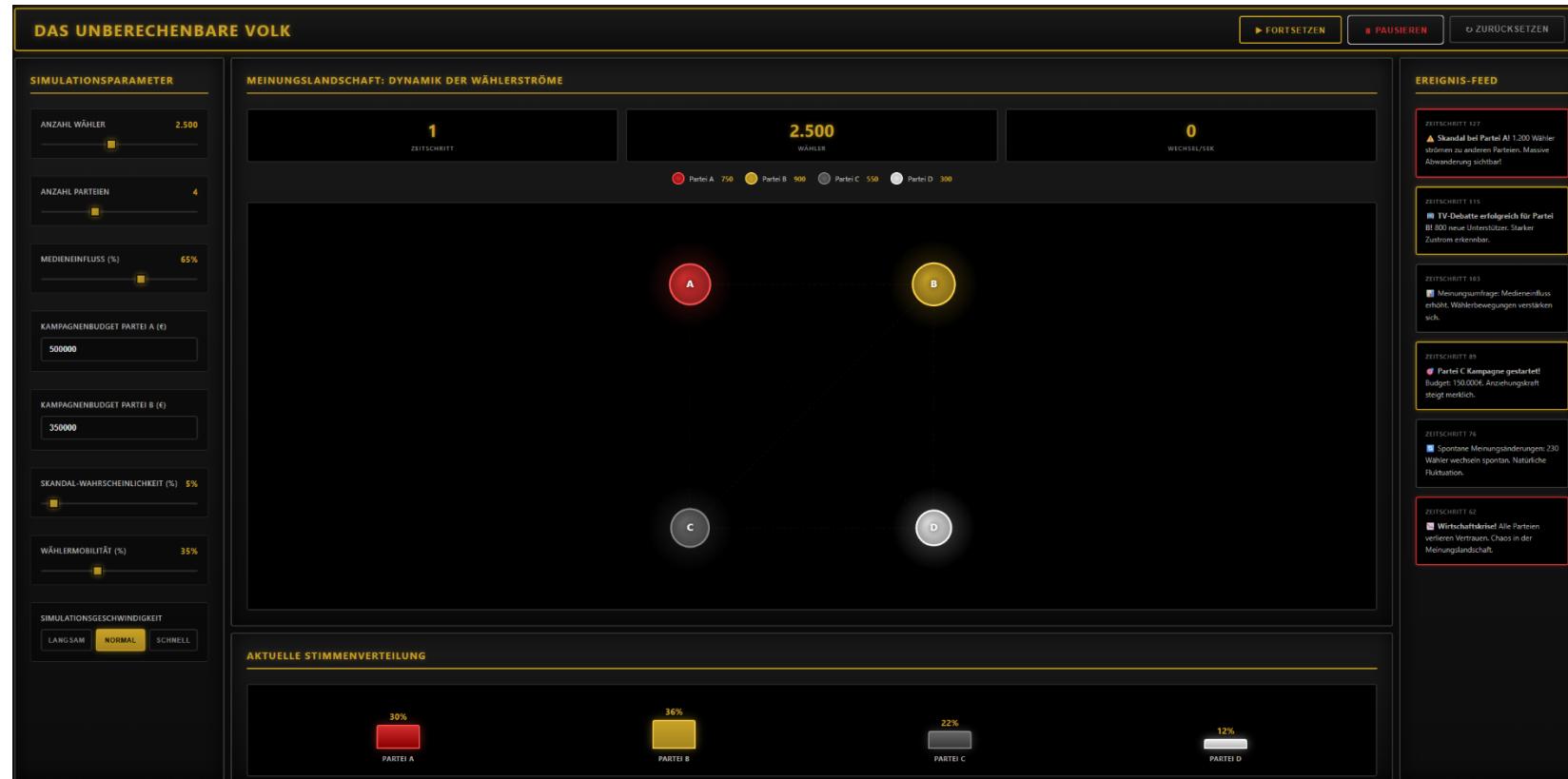


Abbildung 2: Mockup Hauptseite



## A3.2 Statistikseite

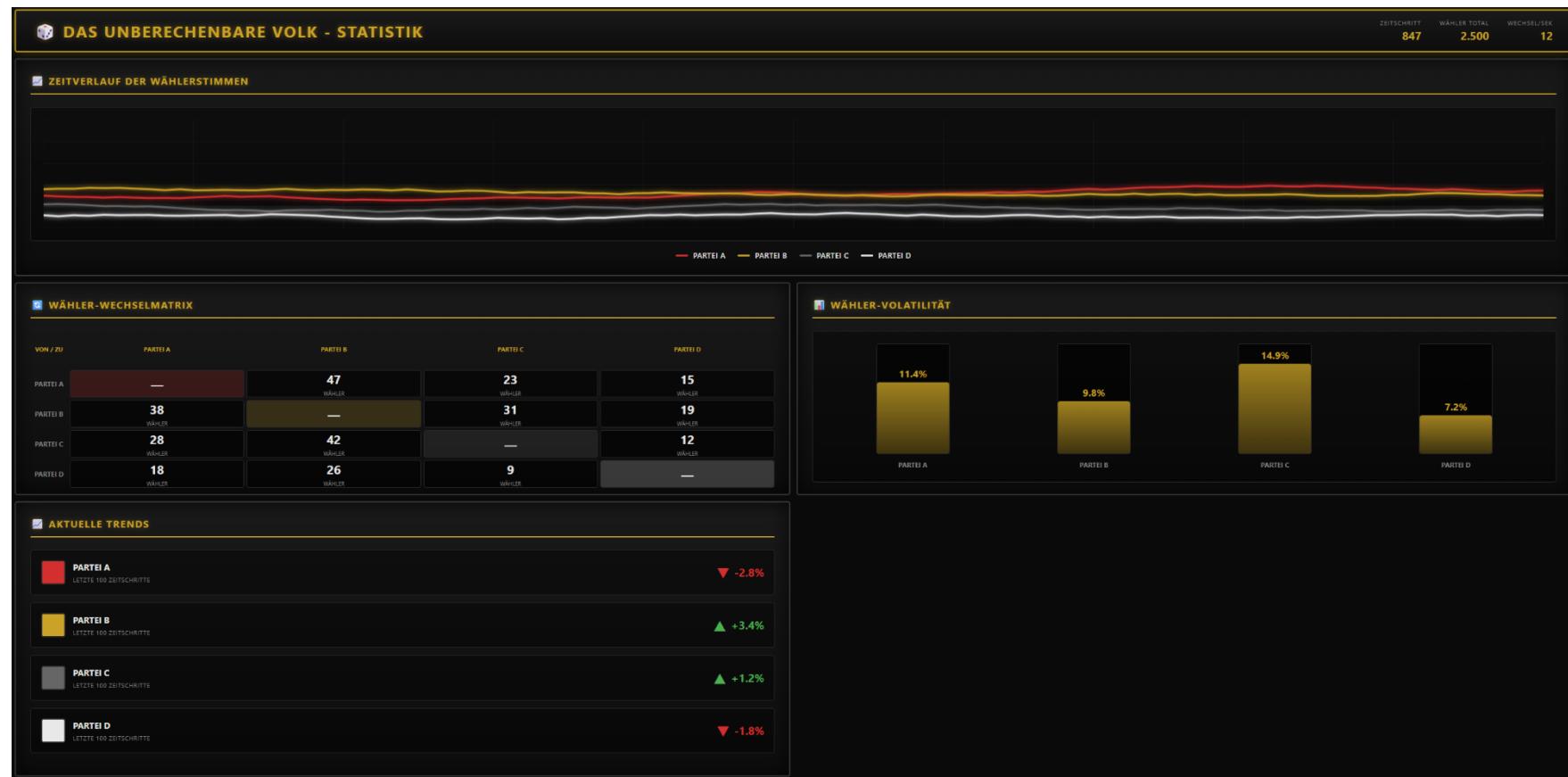


Abbildung 3: Mockup Statistikseite



### A3.3 Parlamentsseite

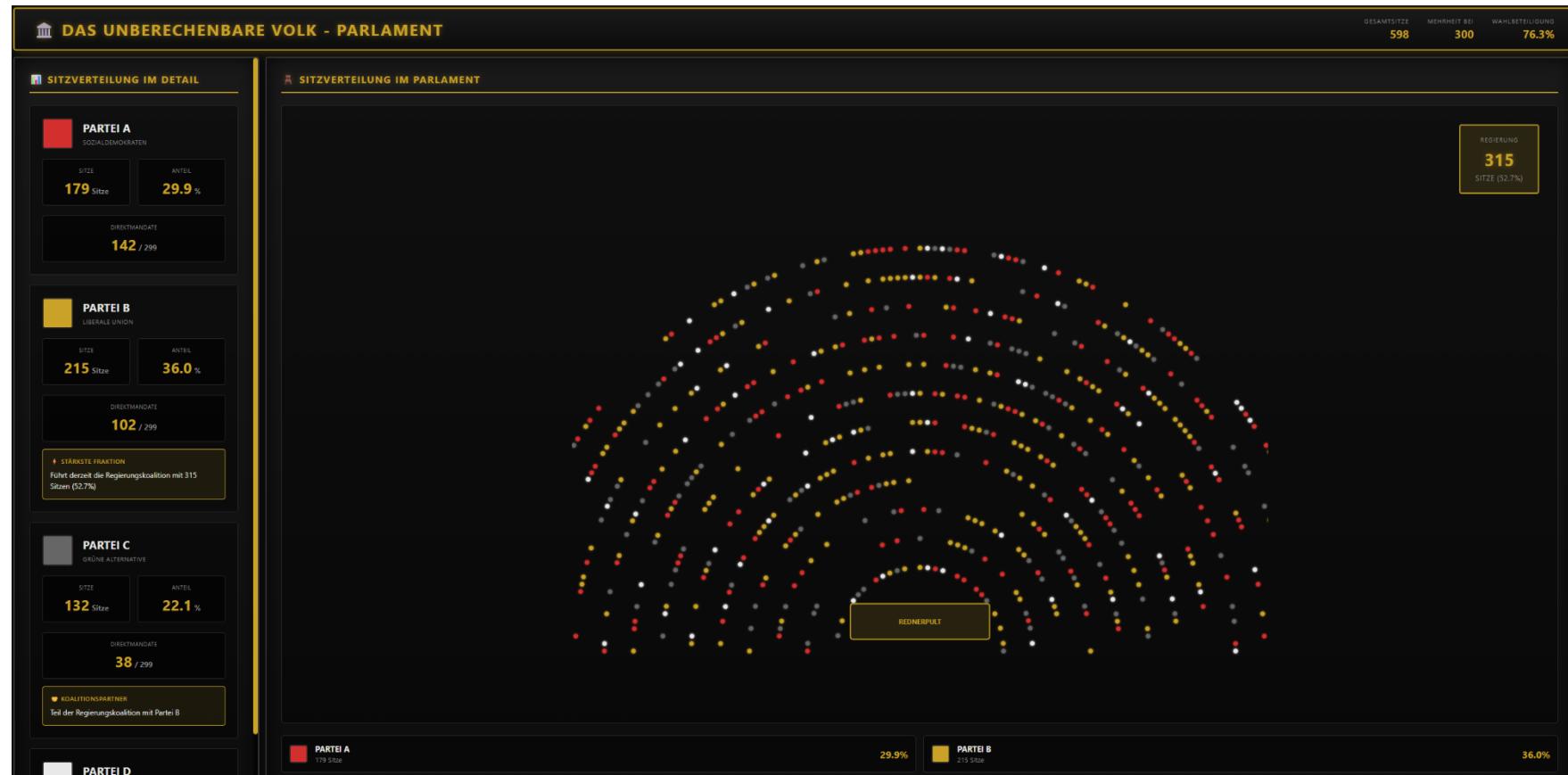


Abbildung 4: Mockup Parlamentsseite



## 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 12: 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 13: 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 14: Nutzwertanalyse GUI-Framework



## A5 Technische Details

### A5.1 Use-Case-Diagramm

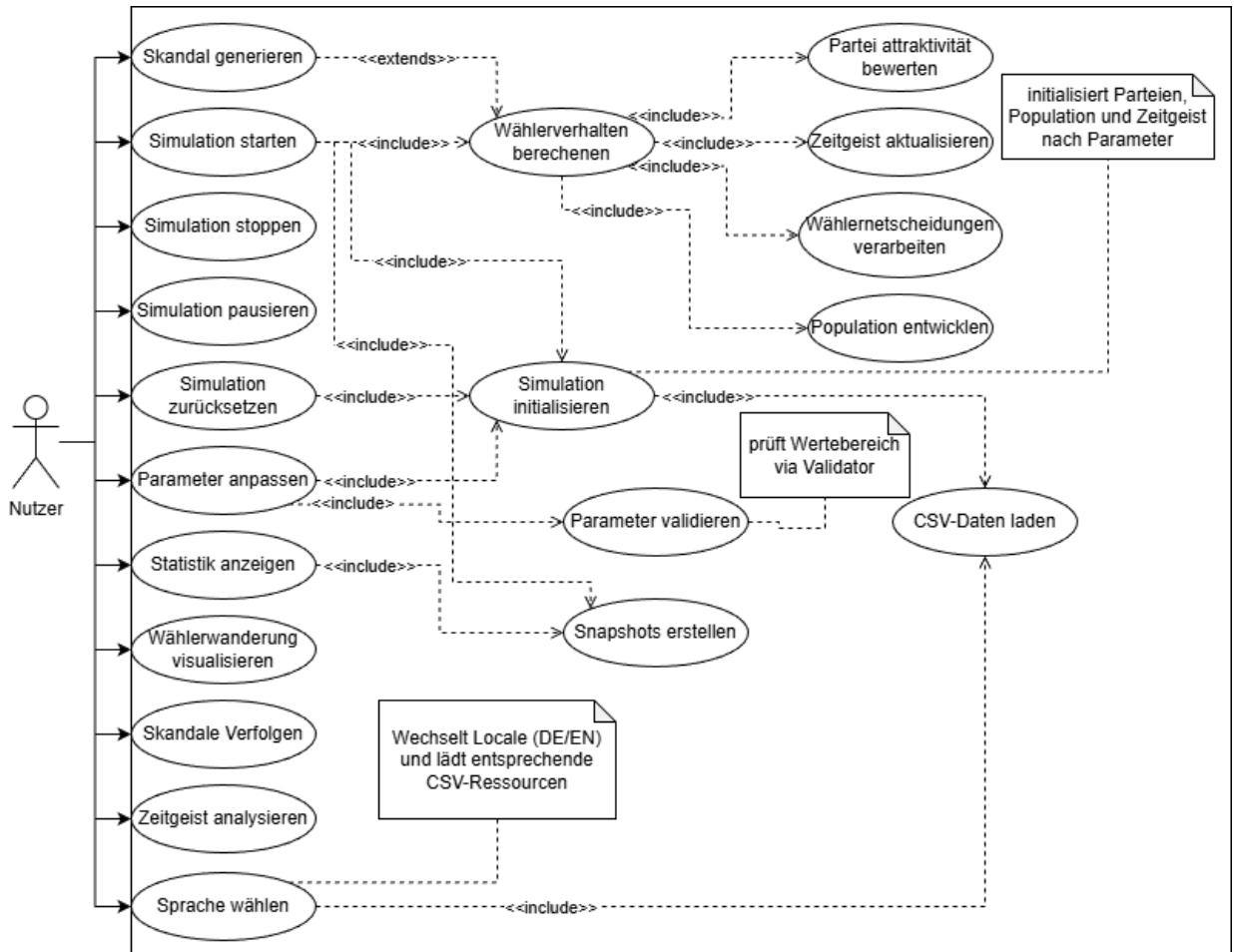


Abbildung 5: Use-Case-Diagramm



## A5.2 Paketdiagramm

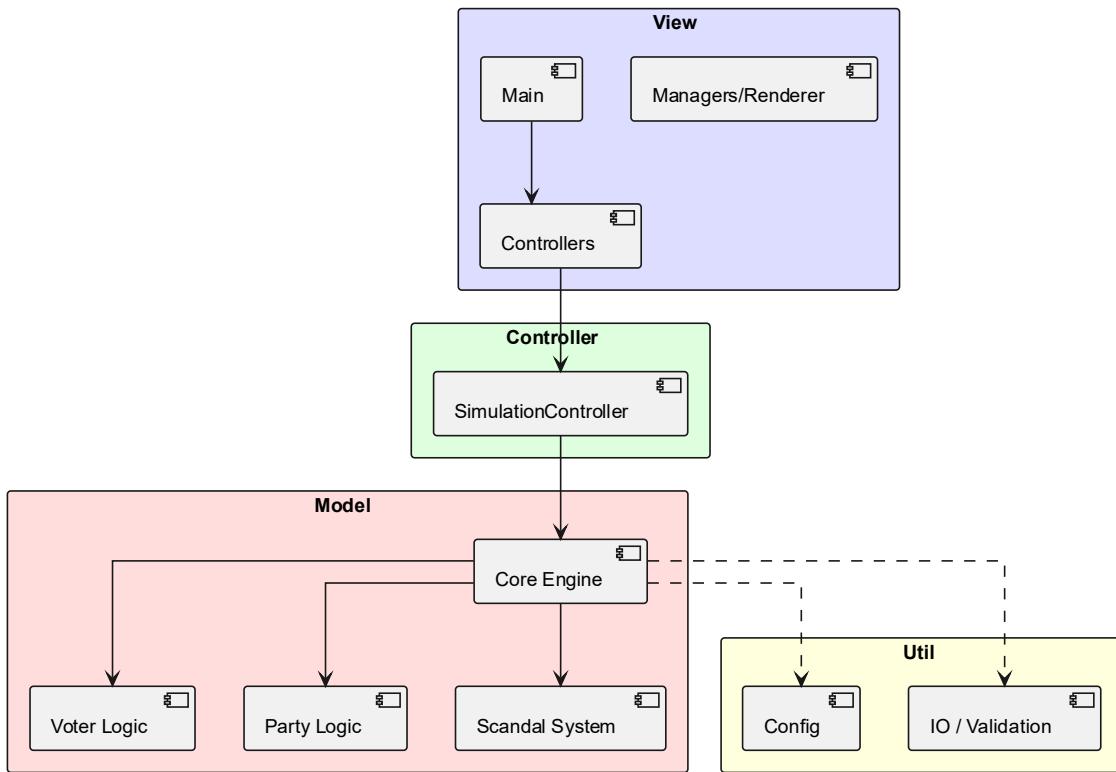


Abbildung 6: Paketdiagramm



### A5.3 Klassendiagramm Model-Layer

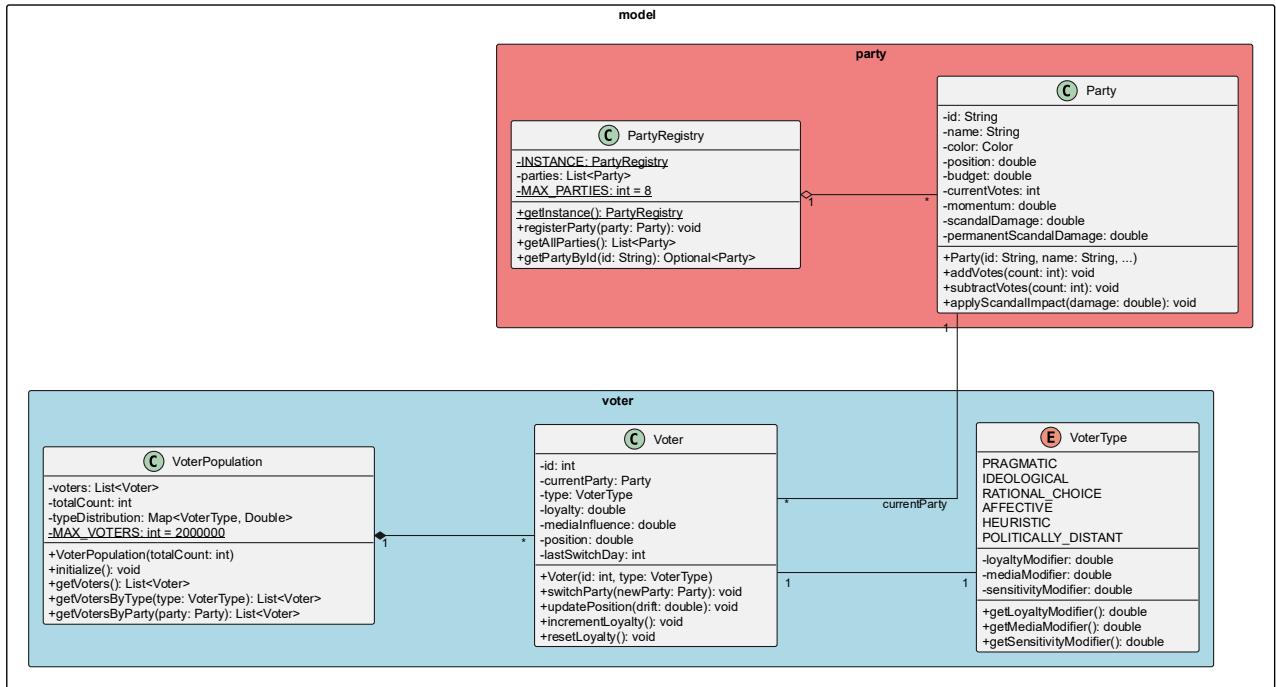


Abbildung 7: Klassendiagramm Model-Layer

### A5.4 Klassendiagramm Kalkulationslogik

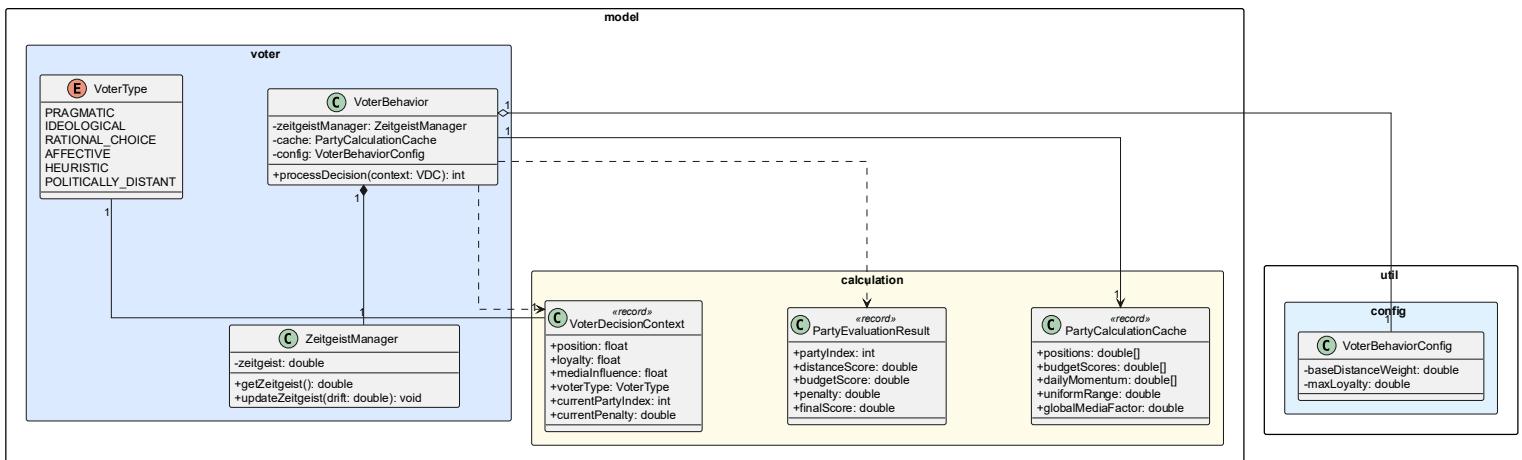


Abbildung 8: Klassendiagramm Kalkulation



## A5.5 Klassendiagramm Core

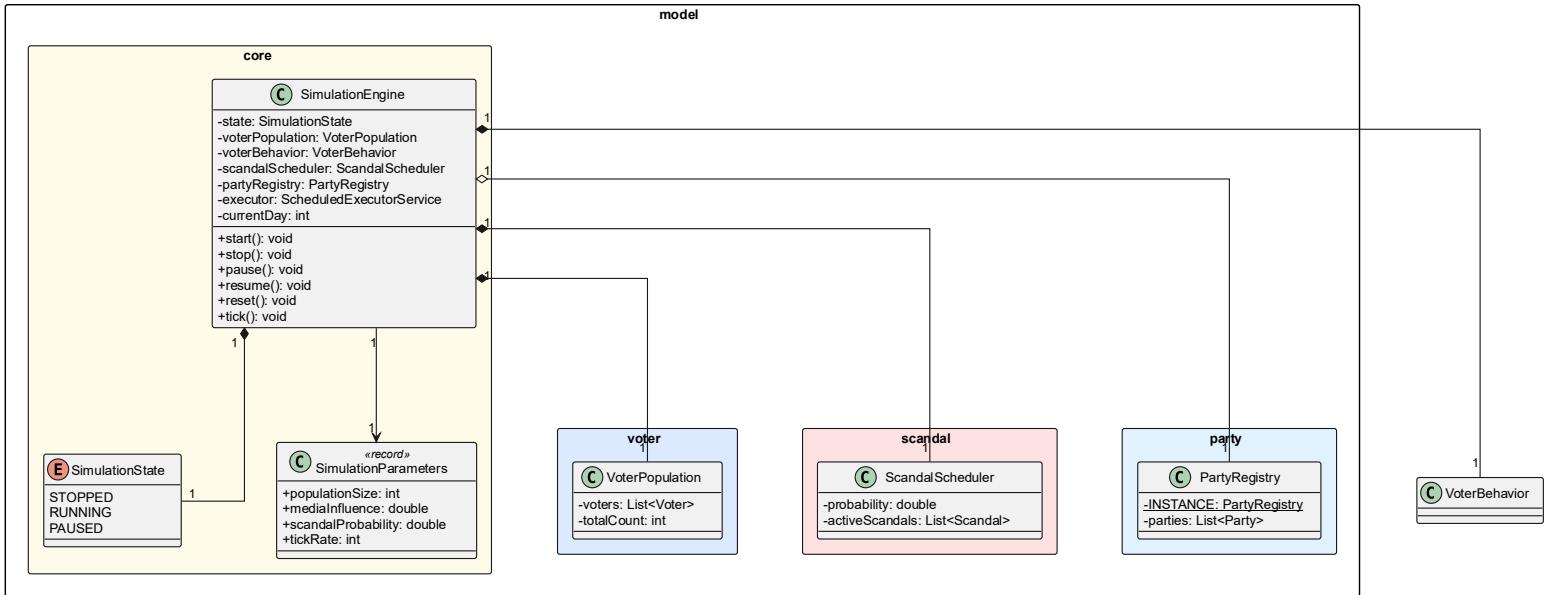


Abbildung 9: Klassendiagramm Core

## A5.6 Klassendiagramm Skandale

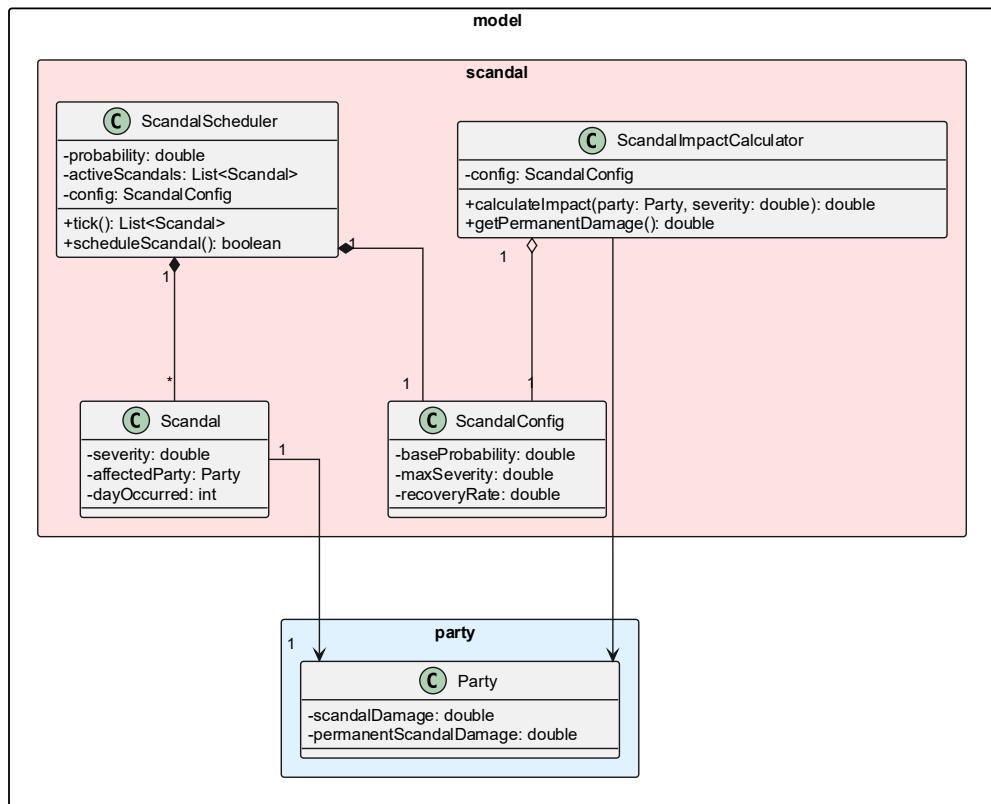


Abbildung 10: Klassendiagramm Skandal



## A5.7 Klassendiagramm View Komponenten

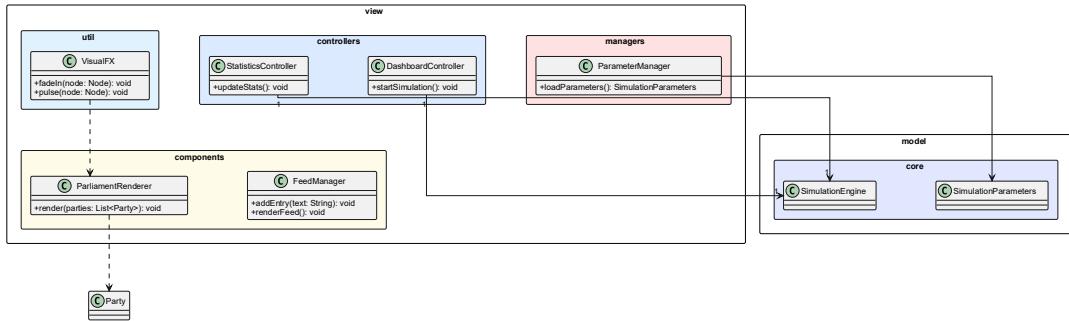


Abbildung 11: Klassendiagramm View Komponenten



## A6 Unit-Tests

Tes t- ID	Testklasse	Testmethode	Beschreibung	Stat us
UT-01	DistributionProviderTes t	testNormalDistribution()	Prüft statistische Eigenschaften der Normalverteilung ( $\mu=0, \sigma=1$ )	<input checked="" type="checkbox"/>
UT-02	DistributionProviderTes t	testExponentialDistributio n()	Validiert Exponentialverteilung mit $\lambda=1.0$	<input checked="" type="checkbox"/>
UT-03	DistributionProviderTes t	testUniformDistribution()	Testet Gleichverteilung im Intervall	<input checked="" type="checkbox"/>
UT-04	ParameterValidatorTest	testValidVoterCount()	Prüft gültige Wähleranzahl (1-2.000.000)	<input checked="" type="checkbox"/>
UT-05	ParameterValidatorTest	testInvalidVoterCountZero ()	Validierung: 0 Wähler → Exception	<input checked="" type="checkbox"/>
UT-06	ParameterValidatorTest	testInvalidVoterCountNega tive()	Validierung: Negative Wähler → Exception	<input checked="" type="checkbox"/>
UT-07	ParameterValidatorTest	testInvalidVoterCountToo High()	Validierung: > 2.000.000 Wähler → Exception	<input checked="" type="checkbox"/>



UT-08	ParameterValidatorTest	testValidPartyCount()	Prüft gültige Parteienanzahl (2-8)	<input checked="" type="checkbox"/>
UT-09	ParameterValidatorTest	testInvalidPartyCountTooFew()	Validierung: < 2 Parteien → Exception	<input checked="" type="checkbox"/>
UT-10	ParameterValidatorTest	testInvalidPartyCountTooMany()	Validierung: > 8 Parteien → Exception	<input checked="" type="checkbox"/>
UT-11	ParameterValidatorTest	testValidBudget()	Prüft gültige Budgets ( $\geq 0$ )	<input checked="" type="checkbox"/>
UT-12	ParameterValidatorTest	testInvalidBudgetNegative()	Validierung: Negatives Budget → Exception	<input checked="" type="checkbox"/>
UT-13	VoterBehaviorTest	testOpinionChange()	Testet Meinungsänderung eines Wählers	<input checked="" type="checkbox"/>
UT-14	VoterBehaviorTest	testVoterTypeInfluence()	Validiert Einfluss des Wählertyps auf Entscheidung	<input checked="" type="checkbox"/>
UT-15	VoterBehaviorTest	testMediaInfluenceEffect()	Prüft Auswirkung von Medieneinfluss	<input checked="" type="checkbox"/>
UT-16	ScandalImpactCalculatorTest	testScandalImpactCalculator()	Berechnet Skandal-Auswirkungen auf Stimmen	<input checked="" type="checkbox"/>
UT-17	ScandalImpactCalculatorTest	testScandalDecayOverTime()	Testet exponentielles	<input checked="" type="checkbox"/>



			Abklingen eines Skandals	
--	--	--	-----------------------------	--

Tabelle 15: Unit-Tests



## A7 Testprotokoll

# Testprotokoll: Blackbox-Tests

<b>Projekt</b>	Das unberechenbare Volk
<b>Version</b>	1.0
<b>Testdatum</b>	03.02.2026
<b>Tester</b>	Nico Hoffmann
<b>Testumgebung</b>	JDK 21, Maven, JavaFX, Schulrechner

---

## 1. Testziele

Verifizierung der funktionalen Anforderung der Wahlsimulation durch systematische Blackbox-Tests ohne Kenntnis der internen Implementierung. Schwerpunkte:

Benutzerinteraktion, Parametervalidierung, Simulationskorrektheit und Systemstabilität.

## 2. Testgegenstand

Das unberechenbare Volk ist eine JavaFX-Anwendung zur Simulation von Wählerverhalten mit folgenden Kernfunktionen:

- Partikel-basierte Visualisierung von Wählerwanderungen
- Echtzeitparameter Anpassung
- Ereignissystem mit automatischen Skandalen
- Live-Statistiken und Graphen
- 6 wissenschaftlich fundierte Wählertypen



### 3. Testfälle

TC-01: Parameteranpassung Medieneinfluss	
<b>Vorbedingung</b>	Simulation läuft
<b>Testeingabe</b>	Medieneinfluss auf 90% setzen
<b>Erwartetes Ergebnis</b>	Parameter akzeptiert, erhöhte Wählerbewegung sichtbar
<b>Tatsächliches Ergebnis</b>	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Fehlgeschlagen
<b>Priorität</b>	Mittel

TC-02: Grenzwerttest Loyalität Minimum	
<b>Vorbedingung</b>	Simulation läuft
<b>Testeingabe</b>	Loyalität = 0%
<b>Erwartetes Ergebnis</b>	Parameter akzeptiert, maximale Wählermobilität
<b>Tatsächliches Ergebnis</b>	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Fehlgeschlagen
<b>Priorität</b>	Mittel

TC-03: Grenzwerttest Skandalrate Maximum	
<b>Vorbedingung</b>	Simulation läuft
<b>Testeingabe</b>	Skandalwahrscheinlichkeit = 60%
<b>Erwartetes Ergebnis</b>	Häufige Skandalmeldung
<b>Tatsächliches Ergebnis</b>	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Fehlgeschlagen
<b>Priorität</b>	Mittel



<b>TC-04: Skandalsystem Funktionalität</b>	
<b>Vorbedingung</b>	Skandalrate > 20%
<b>Testeingabe</b>	Simulation 60 Sekunden beobachten
<b>Erwartetes Ergebnis</b>	Mind. 1 Skandal im Ticker, betroffene Partei verliert Stimmen
<b>Tatsächliches Ergebnis</b>	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Fehlgeschlagen
<b>Priorität</b>	Hoch
<b>TC-05: Langzeitstabilität</b>	
<b>Vorbedingung</b>	Standardparameter
<b>Testeingabe</b>	Simulation 5 Minuten laufen lassen
<b>Erwartetes Ergebnis</b>	Keine Abstürze, konstante Performance
<b>Tatsächliches Ergebnis</b>	<input checked="" type="checkbox"/> Bestanden <input type="checkbox"/> Fehlgeschlagen
<b>Priorität</b>	Hoch

## 4. Zusammenfassung

Kategorie	Anzahl	Bestanden	Fehlgeschlagen
Funktionale	2	2	0
Grenzwerte	2	2	0
Performance	1	1	0
<b>Gesamt</b>	<b>5</b>	<b>5</b>	<b>0</b>

**Kritische Fehler:**  Keine  Anzahl: \_\_\_\_\_



## A8 Finale Anwendung

### A8.1 Start-Screen

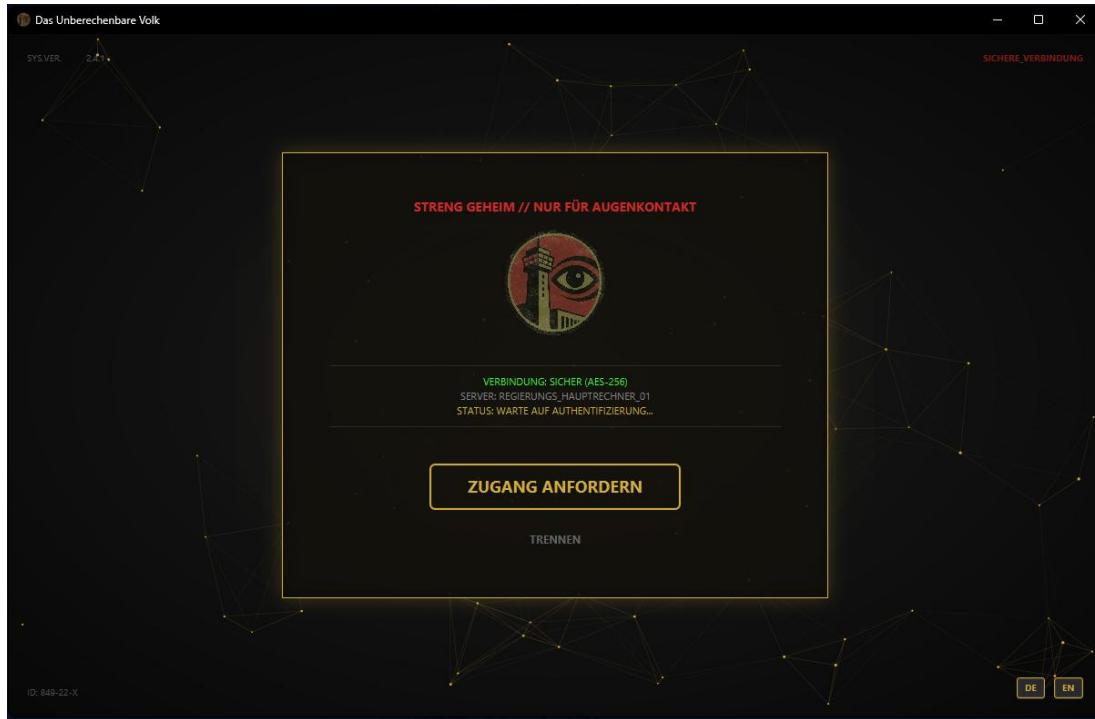


Abbildung 12: Start-Screen



## A8.2 Main-Screen

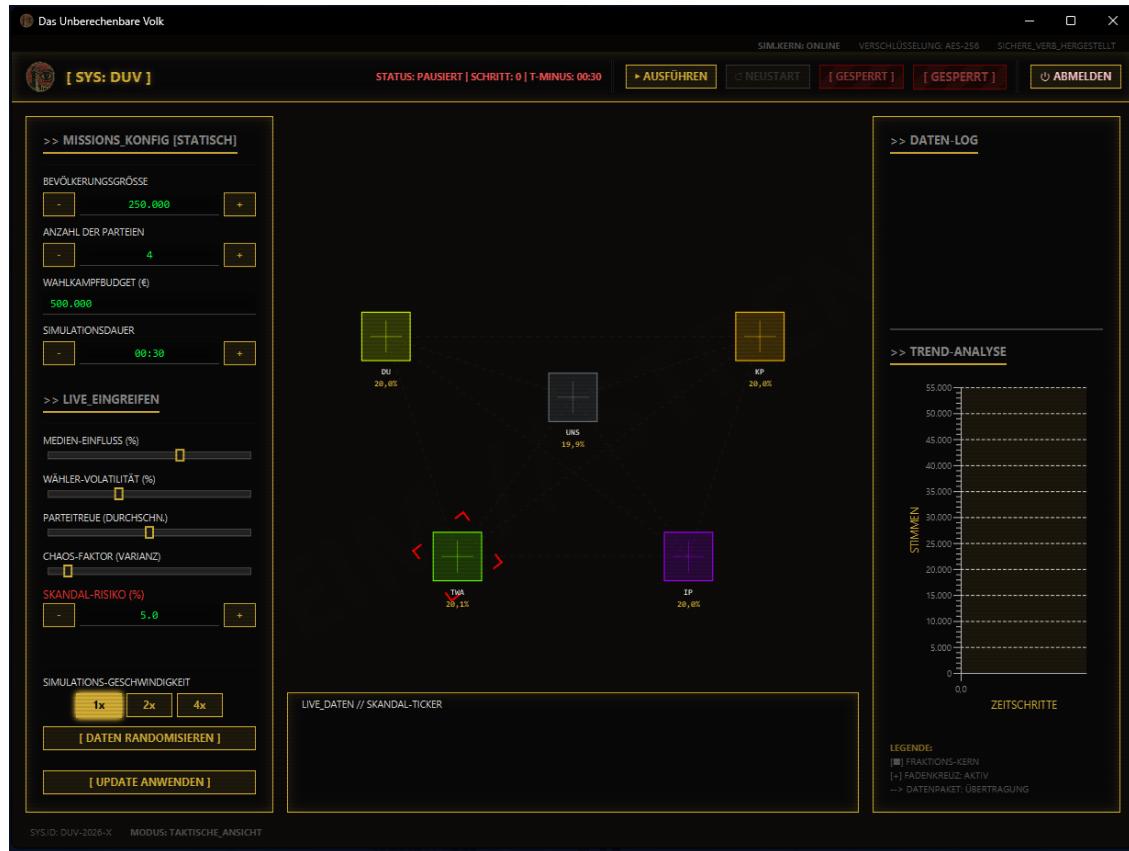


Abbildung 13: Main-Screen



### A8.3 Statistik-Screen



Abbildung 14: Statistik-Screen



#### A8.4 Parlament-Screen

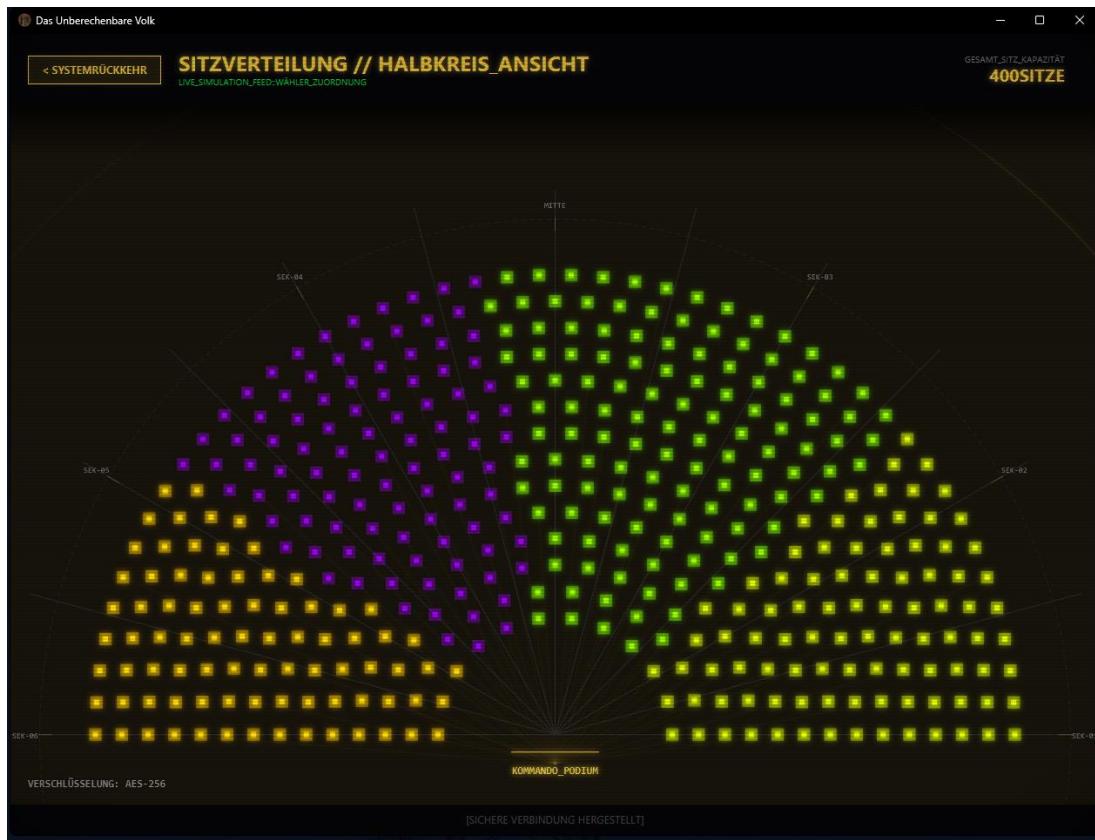


Abbildung 15: Parlament-Screen



## A9 Listings

### A9.1 Kernschleife der SimulationEngine

```
public List<VoterTransition> runSimulationStep() {
    state.incrementStep();

    state.getActiveScandals().removeIf(e ->
state.getCurrentStep() - e.occurredAtStep() >
SimulationConfig.SCANDAL_MAX_AGE_TICKS);

    if (scandalScheduler.shouldScandalOccur() &&
partyRegistry.getParties().size() > 1) {
        triggerNewScandal();
    }

    zeitgeistManager.updateZeitgeist();

    voterBehavior.evolvePopulation(voterPopulation,
parameters);

    double[] acutePressures =
impactCalculator.calculateAcutePressure(
        state.getActiveScandals(),
        partyRegistry.getParties(),
        state.getCurrentStep()
    );

    impactCalculator.processRecovery(partyRegistry.getParties(),
parameters.populationSize());

    List<VoterTransition> transitions =
voterBehavior.processVoterDecisions(
        voterPopulation,
        partyRegistry.getParties(),
        parameters,
        acutePressures,
        impactCalculator,
        zeitgeistManager.getCurrentZeitgeist()
    );

    recalculateCounts();

    return transitions;
}
```

*Listing 1: Kernschleife der SimulationEngine*



## A9.2 Entscheidungslogik in VoterBehavior

```
private double calculateSwitchProbability(VoterDecisionContext context,
SimulationParameters params) {
    double baseMobility = params.volatilityRate() / 100.0;

    double switchProb = baseMobility *
        context.voterType().getLoyaltyModifier() *
        (1.0 - context.loyalty() / VoterBehaviorConfig.LOYALTY_DAMPING_FACTOR) *
        context.mediaInfluence() *
        context.voterType().getMediaModifier();

    if (context.currentPenalty() > 0) {
        double acuteScandalImpact = Math.min(context.currentPenalty() *
            VoterBehaviorConfig.ACUTE_SCANDAL_MULTIPLIER,
            VoterBehaviorConfig.MAX_ACUTE_SCANDAL_BOOST);
        switchProb += acuteScandalImpact;
    }

    if (context.currentPartyIndex() == 0) {
        switchProb *= VoterBehaviorConfig.UNDECIDED_MOBILITY_BONUS;
    }

    return Math.min(switchProb, VoterBehaviorConfig.MAX_SWITCH_PROBABILITY);
}
```

Listing 2:Entscheidungslogik



### A9.3 Generierung von Skandal-Events

```
private void triggerNewScandal() {  
    List<Party> realParties =  
partyRegistry.getTargetableParties();  
  
    if (!realParties.isEmpty()) {  
        int index =  
distributionProvider.getRandomGenerator().nextInt(realParties.  
size());  
        Party target = realParties.get(index);  
        Scandal s = csvLoader.getRandomScandal();  
  
        ScandalEvent event = new ScandalEvent(s, target,  
state.getCurrentStep());  
        state.addScandal(event);  
        target.incrementScandalCount();  
    }  
}
```

*Listing 3: Generierung Skandal-Event*

### A9.4 Abklingalgorithmus der Exponentialverteilung

```
public void processRecovery(List<Party> parties, int totalVoters) {  
    for (int i = 1; i < parties.size(); i++) {  
        if (isValidIndex(i) && partyPermanentDamage[i] > 0) {  
            Party p = parties.get(i);  
            double voterShare = (double) p.getCurrentSupporterCount() / Math.max(1,  
totalVoters);  
  
            double recoveryRate = ScandalConfig.calculateRecoveryRate(voterShare);  
  
            partyPermanentDamage[i] -= recoveryRate;  
            if (partyPermanentDamage[i] < 0) {  
                partyPermanentDamage[i] = 0;  
            }  
        }  
    }  
}
```

*Listing 4: Abklingalgorithmus*



## A9.5 Eingabevalidierung

```
public static String getValidationError(SimulationParameters params) {  
  
    if (params.populationSize() < MIN_POPULATION || params.populationSize() >  
        MAX_POPULATION) {  
  
        return  
ValidationMessage.POPULATION_OUT_OF_RANGE.format(MIN_POPULATION,  
MAX_POPULATION);  
  
    }  
  
  
    if (params.mediaInfluence() < MIN_PERCENTAGE || params.mediaInfluence() >  
        MAX_PERCENTAGE) {  
  
        return  
ValidationMessage.MEDIA_INFLUENCE_OUT_OF_RANGE.format(MIN_PERCENTAGE,  
MAX_PERCENTAGE);  
  
    }  
  
  
    if (params.partyCount() < MIN_PARTIES || params.partyCount() > MAX_PARTIES) {  
  
        return ValidationMessage.PARTY_COUNT_OUT_OF_RANGE.format(MIN_PARTIES,  
MAX_PARTIES);  
  
    }  
  
  
    // Weitere Validierungen für Volatilität, Skandalwahrscheinlichkeit und Budget  
  
    return "";  
}
```

*Listing 5: Eingabevalidierung*



## A9.6 Daten-Update der Diagramme

```
public void update(List<Party> parties, int step) {
    if (historyChart == null || step % UPDATE_INTERVAL != 0) return;

    for (Party p : parties) {
        if (p.getName().equals(SimulationConfig.UNDECIDED_NAME)) continue;

        XYChart.Series<Number, Number> series = historySeriesMap.computeIfAbsent(
            p.getName(),
            ignored -> createSeries(p)
        );

        series.getData().add(new XYChart.Data<>(step, p.getCurrentSupporterCount()));

        if (series.getData().size() > SimulationConfig.HISTORY_LENGTH) {
            series.getData().removeFirst();
        }
    }
}
```

*Listing 6: Daten-Update der Diagramme*



## A9.7 Partikel-Rendering-Loop

```
private void drawParticles() {
    Iterator<MovingVoter> it = activeParticles.iterator();
    while (it.hasNext()) {
        MovingVoter p = it.next();
        p.move();

        double trailLen = 15.0 * currentScaleFactor;
        double angle = Math.atan2(p.y - p.startY, p.x - p.startX);

        if (p.progress < 0.1) {
            angle = Math.atan2(p.targetY - p.startY, p.targetX - p.startX);
        }

        double tailX = p.x - Math.cos(angle) * trailLen;
        double tailY = p.y - Math.sin(angle) * trailLen;

        gc.setStroke(p.color);
        gc.setLineWidth(2.0 * currentScaleFactor);
        gc.strokeLine(tailX, tailY, p.x, p.y);

        gc.setFill(Color.WHITE);
        double headSize = 3.0 * currentScaleFactor;
        gc.fillRect(p.x - headSize / 2, p.y - headSize / 2, headSize, headSize);

        gc.setEffect(new Glow(0.8));

        if (p.hasArrived()) {
            it.remove();
            particlePool.push(p);
        }
    }
    gc.setEffect(null);
}
```

Listing 7: Partikel-Rendering-Loop



**A10 Benutzerhandbuch**

# Benutzerhandbuch

Das unberechenbare Volk





## Inhaltsverzeichnis

<b>1.Einleitung.....</b>	<b>4</b>
<b>2.Kurzbeschreibung des Produktes .....</b>	<b>5</b>
2.1Art des Produktes .....	5
2.2Verwendungszweck.....	5
2.3Zielgruppe .....	5
<b>3.Installationsanleitung.....</b>	<b>6</b>
3.1Systemvoraussetzungen .....	6
3.2Inbetriebnahme der Anwendung .....	6
<b>4.Bedienungsanleitung.....</b>	<b>7</b>
4.1Benutzeroberfläche .....	7
4.2Konfiguration der Parameter.....	7
4.3Statistikoberfläche.....	7
4.4Parlamentoberfläche .....	7
<b>5.Technische Beschreibung.....</b>	<b>8</b>
<b>6.Fehlerbehandlung .....</b>	<b>9</b>
6.1Umgang mit Eingabefehlern .....	9
6.2Technische Störungen .....	9
6.3Protokollierung.....	9
<b>7.Kontaktdaten.....</b>	<b>10</b>
7.1Projektleitung und Entwicklung.....	10
7.2Feedback und Mitwirkung .....	10
<b>8.Glossar .....</b>	<b>11</b>



## 1. Einleitung

### Willkommen zu Das Unberechenbare Volk

Dieses Benutzerhandbuch führt Sie durch die Installation, Bedienung und experimentelle Nutzung der Anwendung: „Das Unberechenbare Volk“ einer interaktiven Simulation von Wählerverhalten, politischen Einflüssen und Skandalen.

Das Handbuch ist so aufgebaut, dass Sie schnell zu Ihrem Ziel kommen: Nach einem Überblick über die Systemanforderungen folgen Kapitel zu Installation, Bedienung, Wartung und Fehlerbehebung. Ein Glossar klärt wichtige Begriffe.

Experimentieren Sie mit Parametern wie Skandalwahrscheinlichkeit oder Budget.

Viel Erfolg bei der Erforschung unberechenbarer Politik!

## 2. Kurzbeschreibung des Produktes

### 2.1 Art des Produktes

Mit „Das Unberechenbare Volk“ wird Ihnen eine interaktive Simulationsumgebung zur Verfügung gestellt, die das dynamische Zusammenspiel zwischen Politik und Wählerschaft abbildet. Das Programm ist als digitale Sandbox konzipiert, in der gesellschaftliche Entwicklungen nicht nur beobachtet, sondern durch gezielte Eingriffe aktiv erforscht werden können.

### 2.2 Verwendungszweck

Bei dieser Anwendung handelt es sich um eine softwarebasierte Modellierung politischer Prozesse. Der Fokus liegt auf der Visualisierung von Ursache-Wirkungs-Prinzipien: Es wird aufgezeigt, wie sich einzelne Ereignisse, seien es gezielte Budgetentscheidungen oder unvorhergesehene Skandale, auf die Stimmung und das Wahlverhalten einer Bevölkerung auswirken. Die Simulation dient dazu, die Volatilität und Komplexität moderner Demokratien greifbar zu machen.



## 2.3 Zielgruppe

Das Programm richtet sich an alle, die Freude am Experimentieren mit Systemen haben:

- **Interessierte Nutzer** erhalten ein Werkzeug, um spielerisch politische Szenarien durchzuspielen und deren Eigendynamik zu beobachten.
- **Im Bildungsbereich** lädt die Simulation dazu ein, komplexe Phänomene wie die Sprunghaftigkeit von Wählern oder das Scheitern an parlamentarischen Hürden nicht nur theoretisch zu betrachten, sondern im Modell unmittelbar zu erleben.
- **Strategen** bietet die Anwendung die Möglichkeit, die Belastbarkeit eines politischen Systems unter extremen Bedingungen auszuloten.

## 3. Installationsanleitung

Um mit der Erforschung politischer Dynamiken in „Das Unberechenbare Volk“ zu beginnen, ist kein klassischer Download-Prozess erforderlich. Die Anwendung wird Ihnen in einer vorbereiteten Ordnerstruktur zur Verfügung gestellt, die neben dem ausführbaren Programm auch den Quellcode und die zugehörige Dokumentation enthält.

### 3.1 Systemvoraussetzungen

Bevor Sie die Anwendung starten, stellen Sie bitte sicher, dass Ihr System die folgenden Voraussetzungen für einen stabilen Simulationsbetrieb erfüllt:

- **Betriebssystem:** Windows 10/11.
- **Prozessor:** Ein moderner Mehrkern-Prozessor, um die parallelen Berechnungen der Wählerpopulation und der Simulationseinflüsse flüssig zu verarbeiten.
- **Arbeitsspeicher:** Mindestens 4 GB RAM, wobei 8 GB für eine verzögerungsfreie Darstellung der komplexen Grafik-Partikel und Statistiken empfohlen werden.
- **Grafik:** Eine Grafikeinheit mit Unterstützung für Hardware-Beschleunigung (DirectX oder OpenGL), um die taktischen Ansichten und Live-Diagramme optimal darzustellen.
- **Bildschirm:** Eine Auflösung von mindestens 1280x720 Pixeln, damit alle Steuerungselemente der Missions-Konfiguration und der Skandal-Ticker vollständig sichtbar bleiben.



## 3.2 Inbetriebnahme der Anwendung

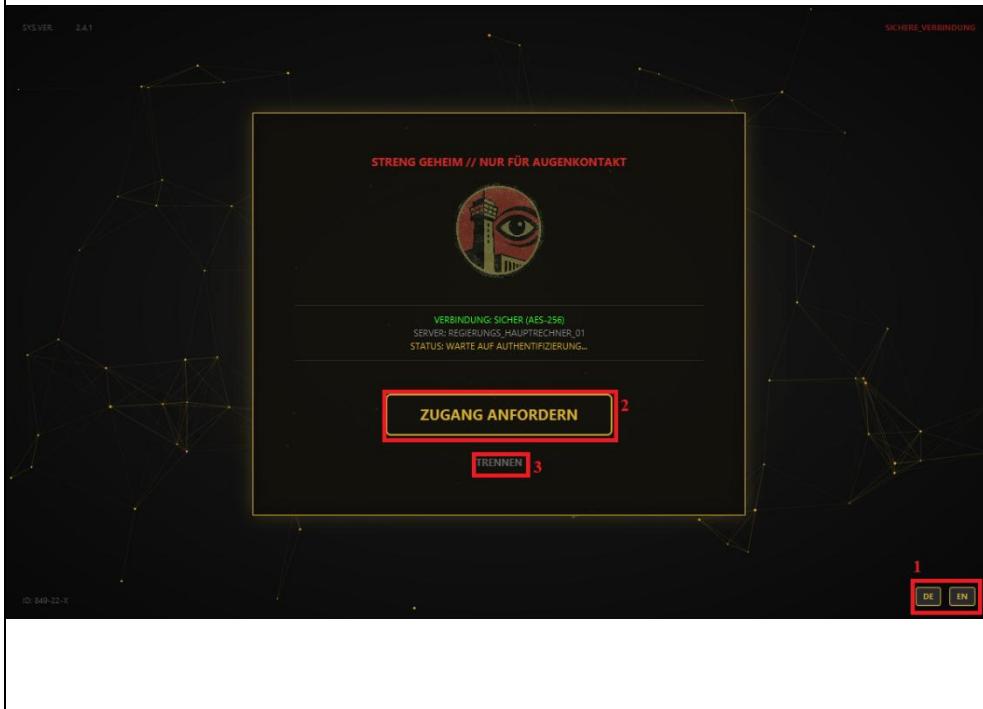
Da die Software als fertiges Paket bereitgestellt wird, führen Sie zur Inbetriebnahme lediglich folgende Schritte aus:

1. **Verzeichnis öffnen:** Navigieren Sie in den bereitgestellten Projektordner. Dort finden Sie die Unterordner für den Quellcode (src), die Dokumentation und das Hauptverzeichnis der Anwendung.
2. **Startvorgang:** Suchen Sie im Hauptverzeichnis nach der Datei „DUV.exe“.
3. **Ausführung:** Ein einfacher Doppelklick auf diese Datei genügt, um die Anwendung zu starten. Sollte Ihr System den Direktstart blockieren, kann die Anwendung alternativ über die Konsole mit dem Befehl java -jar DUV.exe gestartet werden.

Sobald der Startbildschirm erscheint, sind Sie bereit für Ihre erste Sitzung im Regierungs-Hauptrechner.

## 4. Bedienungsanleitung

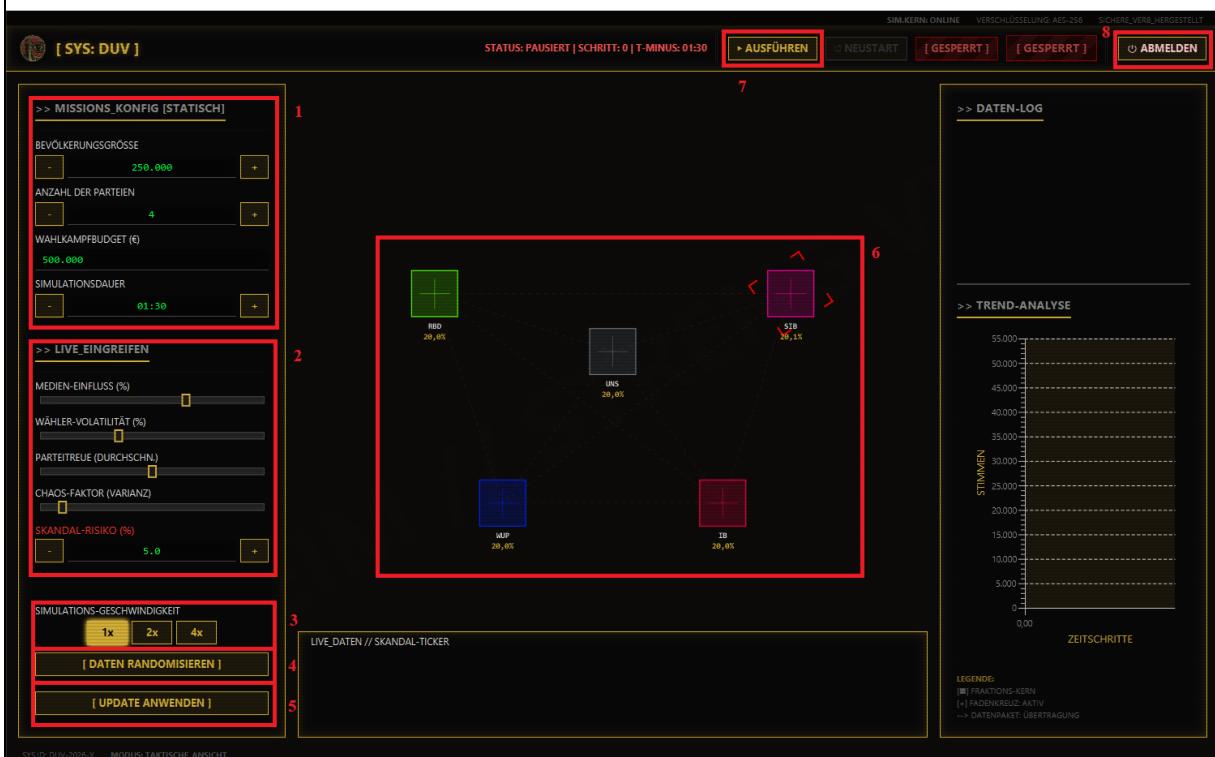
### 4.1 Benutzeroberfläche





- Sprachauswahl:** Schaltet die gesamte Benutzeroberfläche zwischen Deutsch (DE) und Englisch (EN) um
- Zugang-Anfordern:** Der primäre Interaktionspunkt, um die Simulation zu initialisieren und das Dashboard zu laden.
- Verbindung trennen:** Beendet die Anwendung sicher.

## 4.2 Konfiguration der Parameter



### Bereich 1 - MISSIONS\_KONFIG [STATISCH]

- BEVÖLKERUNGSGRÖSSE:** Gesamtzahl der simulierten Wähler (1.000–2.000.000).
- ANZAHL DER PARTEIEN:** Anzahl der konkurrierenden Fraktionen (2–8).
- WAHLKAMPFBUDGET (€):** Gesamtbudget (0–500.000.000), wird ungleichmäßig auf alle Parteien aufgeteilt.
- SIMULATIONSDAUER:** Dauer der Simulation (00:30–05:00).



## Bereich 2 - LIVE\_EINGREIFEN

- **MEDIEN-EINFLUSS (%)**: Auswirkungen von Skandalen
- **WÄHLER-VOLATILITÄT (%)**: Wechselbereitschaft der Bevölkerung. Niedrig = stabil, hoch = sprunghaft
- **PARTEITREUE (DURCHSCHN.)**: Stammwählerbindung. Hohe Werte schützen vor plötzlichen Stimmenverlusten
- **CHAOS-FAKTOR (VARIANZ)**: Zufällige Unvorhersehbarkeit im System. Verhindert rein lineare Entwicklungen
- **SKANDAL-RISIKO (%)**: Wahrscheinlichkeit zum Erscheinen eines Skandals

## Button 3 - SIMULATIONS-GESCHWINDIGKEIT

- **1x / 2x / 4x**: Zeitraffer-Multiplikator

## Button 4 - [ DATEN RANDOMISIEREN ]

- Setzt alle Parameter, außer die Simulationsdauer zufällig

## Button5 - [ UPDATE ANWENDEN ]

- Übernimmt Änderungen der Live-Parameter ohne Neustart

## Animation 6 - Taktische Ansicht

- Visualisierung der Parteien mit aktuellen Stimmenanteilen

## Button 7 - AUSFÜHREN

- Beginnt die Simulation



The screenshot displays the 'SYS: DUV' interface with several key sections:

- Bereich 1 - LIVE\_DATEN // SKANDAL-TICKER**: Shows a live scandal ticker for 'Visa-Affäre' targeting 'EU' with a progress bar at -92%.
- Button 2 - DATEN**: A red-bordered box labeled '2'.
- Button 3 - Parlament**: A red-bordered box labeled '3'.
- Button 4 - Daten-Log**: A red-bordered box labeled '4'.
- Button 5 - TREND-ANALYSE**: A red-bordered box labeled '5'.

**Left Panel (Missions Konfig [STATISCH]):**

- BEVÖLKERUNG: [GESPERRT]
- ANZAHL STIMMEN: [GESPERRT]
- WAHLKURVE: [GESPERRT]
- SIMULATION: [GESPERRT]

**Mitte (Zentrale Statistik):**

- BU: 14,7%
- UWS: 16,7%
- SDK: 21,9%
- JK: 20,5%
- FK: 20,5%

**Rechte Seite (Trend-Analyse):**

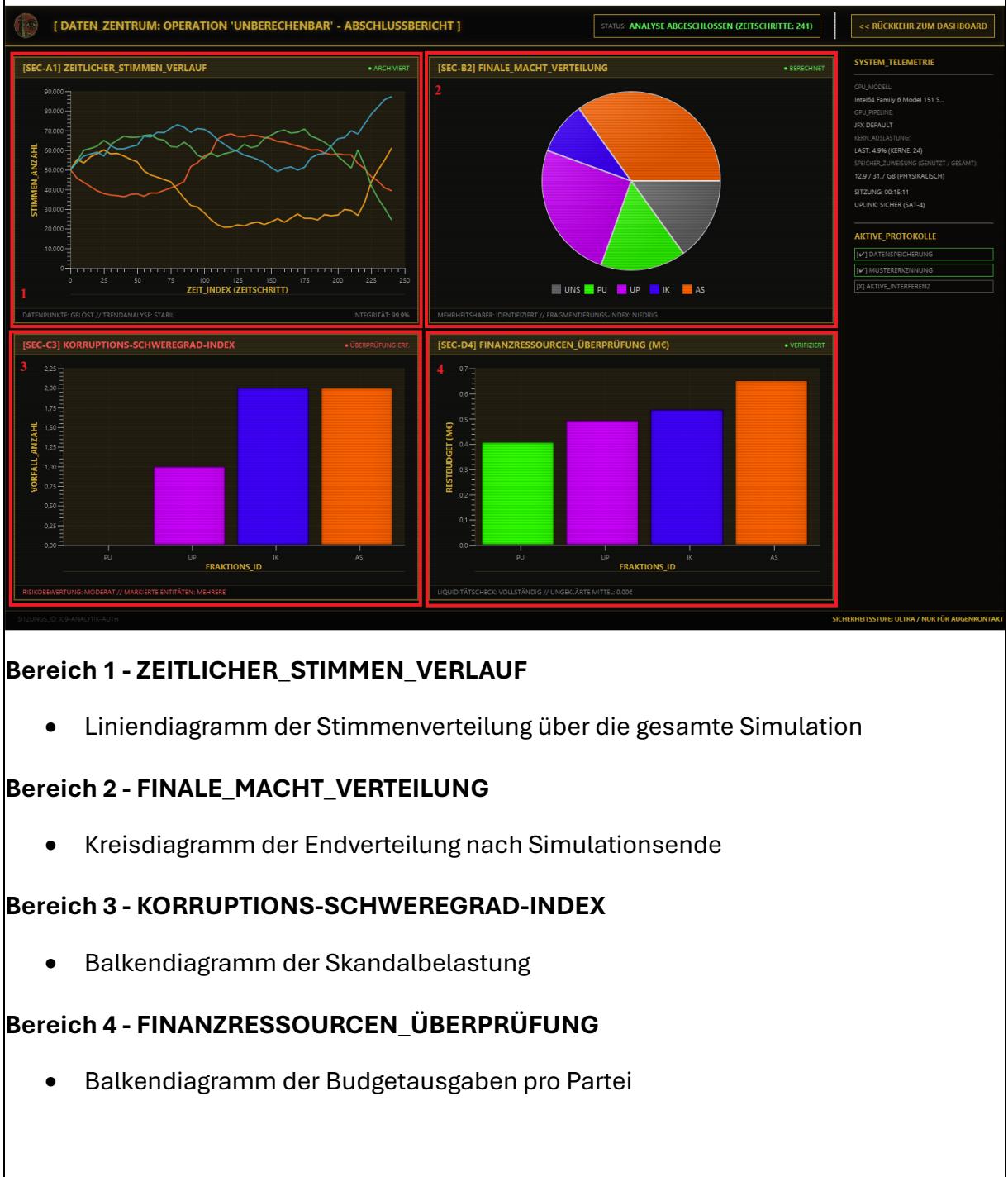
>> TREND-ANALYSE

Diagramm mit Y-Achse 'STIMMEN' (0 bis 90.000) und X-Achse 'ZEITSCHRITTE' (0 bis 125). Die Kurve zeigt die Stimmenverteilung über die Zeitschritte.

LEGENDE:  
[■] FRAKTIONS-KERN  
[+] FÄDENKREUZ-AKTIV  
-> DATENPAKET-ÜBERTRAGUNG

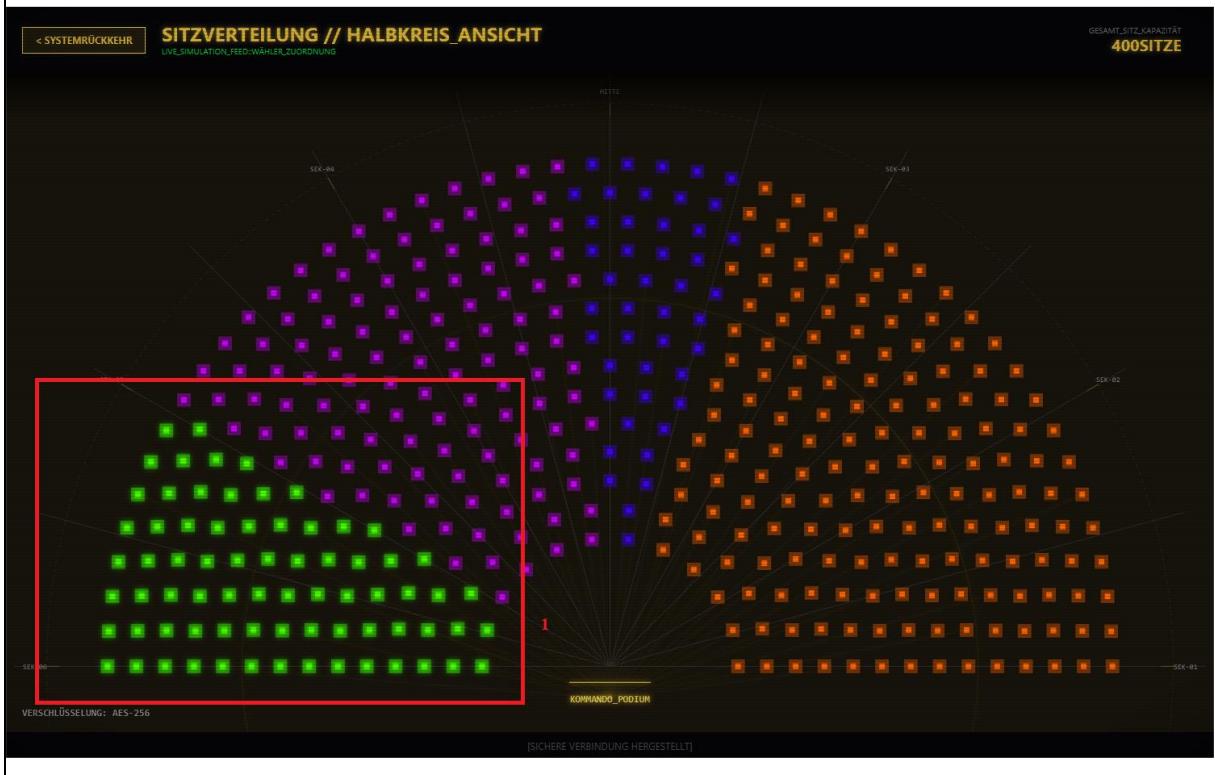


## 4.3 Statistikoberfläche





## 4.4 Parlamentsoberfläche



### Bereich 1 - Parlamentsbereich

- Anzahl der Sitze pro Partei nach Ende der Simulation



## 5. Technische Beschreibung

Dieses Kapitel erklärt, wie die Simulation im Hintergrund funktioniert. Ohne, dass Sie Programmierung verstehen müssen.

### 5.1 Wie arbeitet die Simulation?

„Das Unberechenbare Volk“ läuft vollständig auf Ihrem Computer, ohne Internetverbindung. Alle Berechnungen finden lokal statt – Begriffe wie "Uplink" oder "Server-Verbindung" sind nur Teil des Design-Themas.

Die Simulation besteht aus drei Hauptbereichen:

#### 1. Wähler-Verhalten

- Es werden bis zu 2.000.000 virtuelle Wähler simuliert
- Jeder Wähler gehört einem von 6 wissenschaftlich fundierten Typen
- Diese Typen reagieren unterschiedlich auf Medien, Skandale und Budgets
- In jedem Zeitschritt entscheidet jeder Wähler neu, ob er seiner Partei treu bleibt oder wechselt

#### 2. Parteien-System

- Jede Partei hat ein Budget für Kampagnen
- Höheres Budget macht die Partei attraktiver für Wähler
- Skandale reduzieren die Anziehungskraft einer Partei sofort
- Die Position im politischen Spektrum beeinflusst, welche Wähler sich angezogen fühlen

#### 3. Ereignis-Generator

- Basierend auf dem Skandal-Risiko werden zufällig negative Ereignisse erzeugt
- Diese Events werden aus vorbereiteten Textvorlagen geladen
- Jedes Ereignis hat eine berechnete Auswirkung auf die betroffene Partei



## 5.3 Wo werden die Daten gespeichert?

### Während der Simulation:

- Alle Daten befinden sich im Arbeitsspeicher
- Nichts wird auf der Festplatte gespeichert
- Bei einem Neustart gehen die aktuellen Ergebnisse verloren

### Permanente Daten:

- Parteinamen und Skandal-Vorlagen liegen als Textdateien vor
- Diese befinden sich im Unterordner data/
- Verändern Sie diese Dateien nicht, um Fehler zu vermeiden

## 6. Fehlerbehandlung

Trotz der stabilen Architektur der Simulation können in Ausnahmefällen Unregelmäßigkeiten auftreten. Das System verfügt über integrierte Schutzmechanismen, die Sie durch entsprechende Rückmeldungen bei der Lösung unterstützen.

### 6.1 Umgang mit Eingabefehlern

Sollten bei der Konfiguration der Simulationsparameter Werte eingegeben werden, die außerhalb der logischen Grenzen liegen, verweigert das System die Ausführung.

- **Ungültige Parameter:** Wenn eine Fehlermeldung bezüglich der Populationsgröße oder der Tick-Rate erscheint, prüfen Sie bitte, ob positive Ganzahlen eingegeben wurden.



## 6.2 Technische Störung

Falls die Anwendung nicht wie erwartet reagiert, können folgende Schritte zur Wiederherstellung der Funktionalität führen:

- **System-Reset:** Sollte die Simulation in einen undefinierten Zustand geraten, nutzen Sie die Funktion „Neustart“ im Dashboard. Dies setzt die gesamte SimulationEngine zurück und initialisiert die Wählerpopulation sowie den Zeitgeist neu.
- **Fehlende Daten:** Erscheinen im Skandal-Ticker keine Einträge, stellen Sie sicher, dass die mitgelieferten CSV-Dateien im Datenverzeichnis nicht verschoben oder umbenannt wurden, da diese die Grundlage für die Ereignis-Generierung bilden.
- **Performance-Einbußen:** Bei einer sehr hohen Anzahl an Parteien oder einer großen Wählerpopulation kann die Rechenlast stark ansteigen. Reduzieren Sie in diesem Fall die „Simulations-Geschwindigkeit“ im Dashboard, um die CPU-Auslastung zu stabilisieren.



## 7. Kontaktdaten

Sollten Sie Fragen zur Simulation haben, auf technische Hürden stoßen oder Anregungen für die Weiterentwicklung von „Das Unberechenbare Volk“ teilen wollen, stehen Ihnen verschiedene Kanäle zur Verfügung

### 7.1 Projektleitung und Entwicklung

Verantwortlich für die Konzeption, das Design der Benutzeroberfläche sowie die algorithmische Umsetzung der Simulationslogik ist:

- **Name:** Nico Hoffmann
- **Projektrolle:** Chefentwickler & Systemarchitekt
- **E-Mail:** dev.nhoffmann@proton.me

**Technischer Support** Für Anfragen bezüglich der Quellcode-Struktur oder Unterstützung bei der Komplilierung des Projekts können Sie den Entwickler direkt kontaktieren. Bitte geben Sie bei technischen Problemen stets Ihre Systemkonfiguration (Betriebssystem und RAM) sowie die Art des Vorfalls an.

### 7.2 Feedback und Meinung

Die Erforschung unberechenbarer Systeme lebt vom Austausch. Wenn Sie im Rahmen Ihrer Experimente auf interessante Wählerdynamiken oder ungewöhnliche Parlamentskonstellationen gestoßen sind, freuen wir uns über Ihren Bericht.



## 8. Glossar

In diesem Glossar finden Sie Erläuterungen zu den zentralen Begriffen und Mechanismen der Simulation, um ein tieferes Verständnis der politischen Modellierung zu ermöglichen.

- **Budget-Effektivität:** Dieser Parameter gewichtet die Macht finanzieller Ressourcen im Wahlkampf. Er bestimmt, wie stark Investitionen in die Gunst der Wähler umschlagen.
- **Chaos-Faktor:** Ein einstellbarer Wert, der dem System eine Prise Unvorhersehbarkeit hinzufügt. Er modelliert zufällige Abweichungen und sorgt dafür, dass Entwicklungen nicht rein linear verlaufen.
- **Fraktions-Kern:** Die stabile Basis einer Partei innerhalb der parlamentarischen Darstellung. Er wird in der taktischen Ansicht visualisiert und repräsentiert die Kernwählerschaft einer Fraktion.
- **Loyalitäts-Durchschnitt:** Legt das historische Fundament der Parteibindung fest. Ein hoher Wert bedeutet, dass Wähler weniger anfällig für kurzfristige Störfeuer oder Skandale sind.
- **Medien-Einfluss:** Modelliert die Empfänglichkeit der Wähler für externe Narrative und Berichterstattung. Ein hoher Einfluss verstärkt die Auswirkungen von Skandalen und Trends.
- **Parlamentarische Hürde:** Die rechnerische Schwelle, die eine Partei erreichen muss, um Mandate in der Sitzverteilung des Parlaments zu erhalten.
- **Skandal-Risiko:** Regelt die Frequenz unvorhergesehener Störfeuer. Je höher dieser Wert ist, desto öfter werden Ereignisse generiert, die den „akuten Druck“ auf die betroffenen Parteien erhöhen.
- **Tick-Rate:** Steuert den Zeitraffer bzw. die Geschwindigkeit, mit der die Simulation voranschreitet. Jeder „Tick“ entspricht einem abgeschlossenen Berechnungsschritt der Engine.
- **Volatilitäts-Rate:** Definiert die allgemeine politische Fluktuation im System. Sie gibt an, wie wechselbereit die Bevölkerung grundsätzlich ist.
- **Wähler-Transition:** Der Prozess, bei dem ein Wähler aufgrund von Einflüssen seine Entscheidung ändert und von einer Fraktion zu einer anderen (oder in das Lager der Nichtwähler) wechselt.
- **Zeitgeist:** Ein dynamischer Wert, der die allgemeine gesellschaftliche Stimmungslage darstellt. Er wird regelmäßig aktualisiert und beeinflusst, welche politischen Richtungen aktuell einen Vorteil bei der Wählerwerbung haben.