Bomb.io  
Konzeptbericht (Applikationsentwicklung)

| Auftraggeber | Tomas Fehr |
| --- | --- |
| Projektleiter | Mario Aeberhard |
| Autor | Mario Aeberhard, Loic Tobler, Nicolas Ammeter |
| Klassifizierung | *Nicht klassifiziert, ~~Intern, Vertraulich, GEHEIM~~* |
| Status | *In Arbeit, Genehmigt* |
|  |  |

Änderungsverzeichnis

| Datum | Version | Änderung | Autor |
| --- | --- | --- | --- |
| 16.10.2025 | V1.0 | Initiale Erstellung | Mario Aeberhard,  Loic Tobler,  Nicolas Ammeter |
|  |  |  |  |

**Inhaltsverzeichnis**

1 Zusammenfassung 2

2 Systemanforderungen 2

2.1 Fachliche Entitätstypen 2

2.2 Anwendungsfälle / Product-Backlog 3

2.3 Anwendungsfall *XYZ* 4

3 Benutzerschnittstelle 5

4 Systemarchitektur 6

4.1 Gliederung der Lösung 6

4.2 Schnittstellen 8

5 Qualitätssicherung 9

6 Projektplanung 9

***Hinweis: Alle blau geschriebenen Teile sind zu ersetzen***

*Speichern Sie Ihren Konzeptbericht mit dem Namen "2\_1\_Konzeptbericht\_<Projektname>" und kopieren Sie ihn auf dem Klassenlaufwerk 306 ins Unterverzeichnis "Ergebnisse\2\_Ergebnisse\_Konzeptphase"*

*Aktualisieren Sie Ihren Projektplan und speichern Sie ihn am selben Ort unter dem Namen "2\_2\_Projektplan\_<Projekname>".*

# Zusammenfassung

*Geben Sie hier eine kurze Zusammenfassung des Inhalts dieses Dokumentes in Bezug auf Ihr Projekt. Nennen Sie pro Kapitel die wichtigsten (projektspezifischen) Inhalte! Sätze, die für jedes beliebige Projekt zutreffen würden, können Sie weglassen. Information, die auch im Inhaltsverzeichnis zu finden ist, können Sie weglassen.*

# Systemanforderungen

*Hier analysieren Sie den Problembereich. Unter Problembereich, oder englisch problem domain versteht man im weitesten Sinn das Geschäftsumfeld, in welchem eine Anwendung betrieben wird, z.B. das Bankgeschäft oder die Steuerverwaltung.*

## Fachliche Entitätstypen

*Zum Begriff Entität / Entitätstyp:*

*Eine Entität ist ein „Ding“, welches im Problembereich existiert.*

*Beispiele:*

* *Ein Hypothekarvertrag zwischen Herr Sowieso und der Bank Geizhals & Co.*
* *Der Benutzer-Account des Benutzers Ray Charles bei Facebook*

*Eine Entität wird einem Entitätstyp zugeordnet. Dieser umfasst alle Entitäten mit den gleichen Attributen.*

*Beispiele:*

* *Hypothekarvertrag. Attribute: Schuldner, Summe, Zins, Laufzeit*
* *Benutzer-Account. Attribute: Name, Vorname, Account-Name, Passwort, E-Mail*

*Der Zusammenhang zwischen Entität und Entitätstyp ist derselbe wie zwischen Objekt und Klasse in der objektorientierten Programmierung oder zwischen Zeile und Tabelle in relationalen Datenbanken.*

*Was soll nun hier gezeigt werden?*

*Die Struktur des Problembereichs, soweit sie für das Projekt relevant ist: Welche fachlichen Entitätstypen existieren und wie stehen diese miteinander in Beziehung? Stellen Sie diese Entitätstypen und Beziehungen als* ***UML-Klassendiagramm*** *dar. Jeder Entitätstyp wird als eine Klasse modelliert. Die Klassen enthalten nur Attribute und keine Operationen. Beschreiben Sie wo nötig jeden Entitätstyp kurz.*

*Beispiel:*



*Sehr oft werden die hier identifizierten Entitätstypen später als Tabellen in einer Datenbank realisiert.*

## Anwendungsfälle / Product-Backlog

*Nun werden die Anforderungen, welche Sie bereits in der Initialisierungsphase erfasst haben präzisiert und ergänzt. Dazu stehen Ihnen zwei Varianten zur Verfügung:*

1. *In Form von Use Cases*
2. *In Form von User-Stories in einem Product-Backlog.*

*Im Fall a werden Sie nachfolgend die Anforderungen in Form von Anwendungsfällen (Use Cases) erfassen.*

*Fall b bedeutet, dass Sie sich für die agile Entwicklung nach Scrum entschieden haben. Sie werden nachfolgend ein erstes Product-Backlog erstellen oder, wenn Sie in der Initialiserungsphase schon eines erstellt haben, dieses ausbauen. Das Product-Backlog werden Sie bis in die Einführungsphase hinein weiterziehen. Achten Sie darauf, dass Ihre User Stories genügend detailliert sind, so dass sie für alle Projektbeteiligten eindeutig sind.*

*Es folgen die Hinweise zu den Use Cases (Fall a).*

*Zur Use Case Beschreibung gehört ein Use Case Diagramm nach UML. Dieses zeigt alle Akteure und das System selber. Akteure können Benutzer sein. Akteure können aber auch externe Systeme sein, zu welchen Ihr System eine Schnittstelle hat. Aber Achtung: Der Webserver, auf welchem Ihre Web-Applikation läuft ist kein externes System!*

*Beispiele:*





*Faustregel: Ein Use Case sollte eine in sich möglichst abgeschlossene Aufgabe, welche der Benutzer des Systems durchführen will, beschreiben.*

*Normalerweise wird man für ein Kleinprojekt zwischen drei und ca. acht Use Cases haben. Diese fliessen dann direkt in die Planung ein: Jede Iteration wird einen oder zwei Use Cases zum Ziel haben.*

*Schlechte Beispiele:*

* *Alter angeben 🡺 Zu fein. Ist eher ein Schritt innerhalb des Use Cases Konto eröffnen*
* *Budgetplaner verwenden 🡺 Unklar. Was ist damit gemeint? Was ist das Ziel dieses Use Cases? Neues Budget erstellen, Ausgabe erfassen, …?*

*Gute Beispiele: Siehe obige Use Case Diagramme.*

## Anwendungsfall *XYZ*

*Zu jedem Use Case folgt nun eine Beschreibung des Ablaufs. Im Zentrum stehen die Aktionen der beteiligten Akteure und die Reaktion des Systems auf diese Aktionen. Halten Sie sich für die Beschreibung der Use Cases an die nachfolgende Tabellenvorlage. Nehmen Sie bei der Beschreibung der Anwendungsfälle Bezug auf die Entitätstypen und auf die Benutzerschnittstelle im nächsten Abschnitt*

|  |  |
| --- | --- |
| ***Anwendungsfall XYZ*** | |
| *Kurzbeschreibung* | *Zweck des Anwendungsfalls in ein, zwei Sätzen* |
| *Akteure* | *Welche Akteure und externen Systeme sind beteiligt?* |
| *Vorbedingungen* | *Welche Voraussetzungen müssen erfüllt sein, damit der Anwendungsfall überhaupt ausgeführt werden kann?* |
| *Ablauf* | *Einzelne Schritte des Anwendungsfalls* |
| *Resultat* | *Welche Resultate ergeben sich nach Ablauf des Anwendungsfalls?* |
| *Ausnahmen* | *Welche Ausnahmen (z.B. Fehlersituationen) können im Ablauf des Anwendungsfalles auftreten?* |

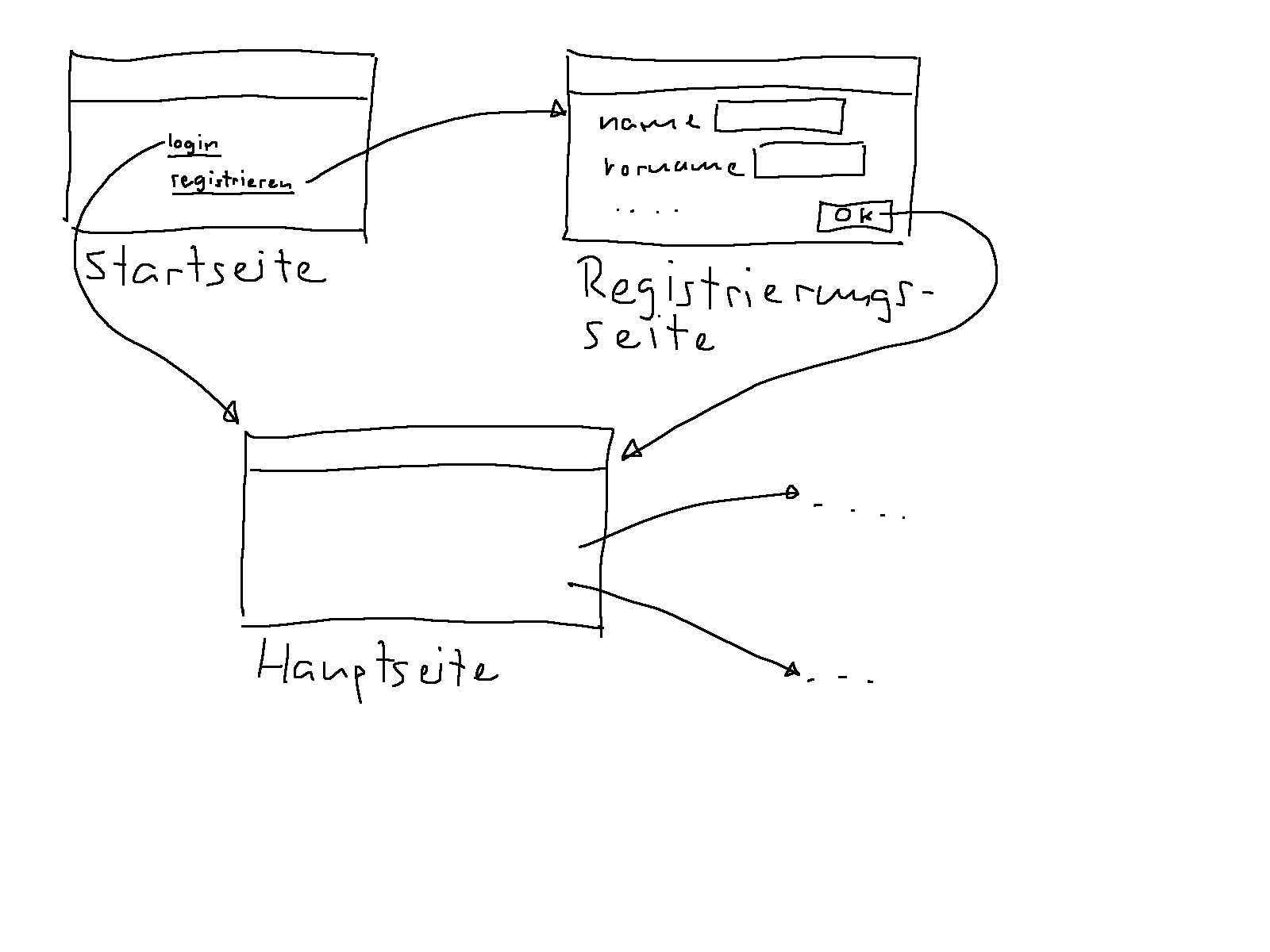
*Beispiel:*

|  |  |
| --- | --- |
| ***Anwendungsfall „Neues Benutzerkonto erstellen“*** | |
| *Kurzbeschreibung* | *Ein Benutzer registriert sich bei der Budgetplaner-Anwendung und erstellt dabei ein neues Benutzerkonto.* |
| *Akteure* | *Benutzer* |
| *Vorbedingungen* | *Der Benutzer hat noch kein Benutzerkonto.* |
| *Ablauf* | 1. *Der Benutzer öffnet im Browser den Startlink der Budgetplaner-Anwendung.* 2. *Das System stellt die Startseite dar.* 3. *Der Benutzer wählt auf der Startseite den Link „Registrieren“* 4. *Das System präsentiert die Registrierungsseite.* 5. *Der Benutzer gibt seinen Namen, seine E-Mail-Adresse, seinen Benutzernamen und zweimal das gleiche Passwort ein.* 6. *Das System validiert die Angaben des Benutzers und erstellt ein neues Benutzerkonto. Anschliessend präsentiert das System die Bestätigungsmeldung auf der Registrierungsseite.* 7. *Der Benutzer kann jetzt zur Startseite zurückkehren oder sich direkt anmelden.* |
| *Resultat* | *Ein neues Benutzerkonto-Objekt mit den Angaben des Benutzers ist erstellt und in der Datenbank gespeichert worden.* |
| *Ausnahmen* | * *Die Validierung der Benutzerangaben in Schritt 6 schlägt fehl.* * *Es besteht bereits ein Benutzer mit diesem Benutzernamen (Schritt 6).* |

# Benutzerschnittstelle

*Skizzieren Sie hier die Benutzerschnittstelle mit Ihren Navigationspfaden. Es geht dabei nicht um eine grafisch und designmässig ausgereifte Darstellung, sondern um den logischen Aufbau. Anhand dieser Skizze müssen die Abläufe in den Use Cases oder den User Stories nachvollzogen werden können. Deshalb müssen alle dort erwähnten Views, Seiten, Dialoge oder Masken hier erscheinen. Die Skizze kann von Hand erstellt und eingescannt werden. Sie können auch mit einem Tool arbeiten.*

*Beispiel:*



# Systemarchitektur

Die gewählte Lösung basiert auf einer browserbasierten Multiplayer-Web-Applikation mit Node.js + Express.js als Server, Socket.IO für die Echtzeitkommunikation und Three.js zur 3D-Darstellung im Browser. Ziel ist ein leichtgewichtiges Spiel ohne Installation, das direkt per URL startet.

## Gliederung der Lösung

**Präsentationsschicht (Client, Browser)**

* **View / Rendering**
  + ThreeScene (Initialisierung der Three.js-Szene)
  + CameraController, Lighting, MapRenderer
  + HUDView (Timer, Bombenanzeige, Spielerstatus)
* **Input & UI**
  + InputController (Tastatur/Maus)
  + Menu/LobbyUI (Name eingeben, Lobby beitreten/erstellen)
* **Netzwerk-Client**
  + SocketClient (Socket.IO-Client, Event-Handling)
* **Client-seitige Logik (thin)**
  + ClientGameController (Statusdarstellung, minimale Vorhersage)
  + **Anti-Cheat (Client-seitig leichtgewichtig):** nur UI-Beschränkungen, Logik bleibt server-autoritätsbasiert

**Applikationsschicht (Server, Node.js)**

* **WebSocket-Gateway**
  + SocketGateway (Annahme von Verbindungen, Namespaces/Rooms)
* **Spiel-Domäne (server-autorität)**
  + MatchmakerService (Lobbys/Rooms)
  + PlayerService (Spieleranmeldung, Zustände)
  + GameLoopService (Tick/Timer, Rundensteuerung)
  + BombService (Bombe zuweisen, Übergaben validieren, Explosion)
  + CollisionService (Berührungsprüfung Server-seitig)
  + RoundService (Rundenstart/-ende, Schwierigkeitseskalation)
  + StateSyncService (Diff/State an Clients senden)
* **Infrastruktur**
  + ExpressApi (statisches Hosting der Client-Assets, Health-Check)
  + Logger
  + (optional später) Metrics
  + **Persistenz**: initial **keine Datenbank** vorgesehen (nur flüchtiger Spielzustand im RAM).

## Schnittstellen

Gemäss Studie/Varianten sind keine externen System-Schnittstellen vorgesehen (keine Konten, keine Fremdsysteme, kein Import/Export). Fokus ist die interne Client↔Server-Echtzeitschnittstelle über Socket.IO (OSI-Layer 7, Nachrichten-basiert).

### Systeminterne Schnittstelle (WebSocket/Socket.IO)

Transport: WebSocket (Socket.IO)  
Rollen: Client (Browser) ↔ Server (Node.js)  
Grundsatz: *Server-Autorität* für alle spielrelevanten Zustände (Positionsvalidierung, Bombenbesitz, Timer).

**Ereignisse (Client → Server)**

1. **join\_lobby**

**Ein Bild, das Text, Schrift, Screenshot, weiß enthält.

KI-generierte Inhalte können fehlerhaft sein.**

**Wirkung:** Server legt Spieler an, ordnet Room zu, antwortet mit lobby\_state/joined.  
**Validierung:** Nickname non-empty, Lobby existent/erstellt.

1. **Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

   KI-generierte Inhalte können fehlerhaft sein.player\_input**

**Wirkung:** Server aktualisiert gewünschte Richtung, speichert seq für ACK/Glättung.

1. **Ein Bild, das Schrift, Text, Reihe, weiß enthält.

   KI-generierte Inhalte können fehlerhaft sein.request\_bomb\_pass**

**Wirkung:** Server prüft Besitz, Kollisions-/Distanzbedingung und Timer, setzt Bombenbesitz um, emittiert bomb\_state.

1. **request\_start**

**Wirkung:** Startet Runde, initialisiert Timer/Bombe(n).

1. **pong (Latenz-Messung)**

****

**Ereignisse (Server → Client)**

1. **joined**

**Ein Bild, das Text, Schrift, weiß, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **lobby\_state**

**Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **game\_state *(periodisch, z. B. 15–30 Hz)***

**Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **bomb\_state**

**Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **explosion**

**Ein Bild, das Text, Schrift, weiß, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **round\_end**

**Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **error**

**Ein Bild, das Text, Schrift, Screenshot, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.**

1. **ping**



# Qualitätssicherung

*Erstellen Sie hier eine erste Tabelle mit den Testfällen für den System und Abnahmetest. Diese werden Sie in den nächsten Phasen ausbauen und genau spezifizieren. Im Moment geht es darum, die Testfälle zu identifizieren. Nehmen Sie als Ausgangspunkt Ihre Use Cases oder User Stories. Als Faustregel gilt: Pro Use Case im Minimum ein Testfall. Wenn Sie mit User Stories arbeiten, wird eine Testfall mehrere User Stories aufs Mal abdecken.*

*Beispiel:*

|  |  |  |
| --- | --- | --- |
| ***Nr.*** | ***Abgedeckter Anwendungsfall oder User Stories*** | ***Beschreibung*** |
| *TC01* | *Neues Benutzerkonto erstellen* | *Ein Testbenutzer, welcher noch kein Konto besitzt, registriert sich beim System und kann sich anschliessend anmelden.* |
| *TC02* | *…* |  |
| *TC03* | *…* |  |

# Projektplanung

*Aktualisieren Sie das Projektplandokument aus der Initialisierungsphase 🡪 neue Version. Machen Sie hier lediglich einen Verweis auf das aktualisierte Dokument*

*Beispiel:*

*Siehe Projektplan, Version 2.0 vom 18.3.2015*

*Im Projektplan selber:*

*Aktualisieren Sie die tatsächlich aufgelaufenen Aufwände für die erledigten Aktivitäten und die geschätzten Aufwände für die kommenden Aktivitäten.*

*Beurteilen Sie auch die Risikosituation erneut. Haben sich an der Risikosituation Änderungen (Verschärfung/Entschärfung, neue Risiken) ergeben? Dann passen Sie die Risikosituation an. Führen Sie wiederum konkrete Massnahmen auf, wie Sie den Risiken in der nächsten Phase begegnen wollen.*