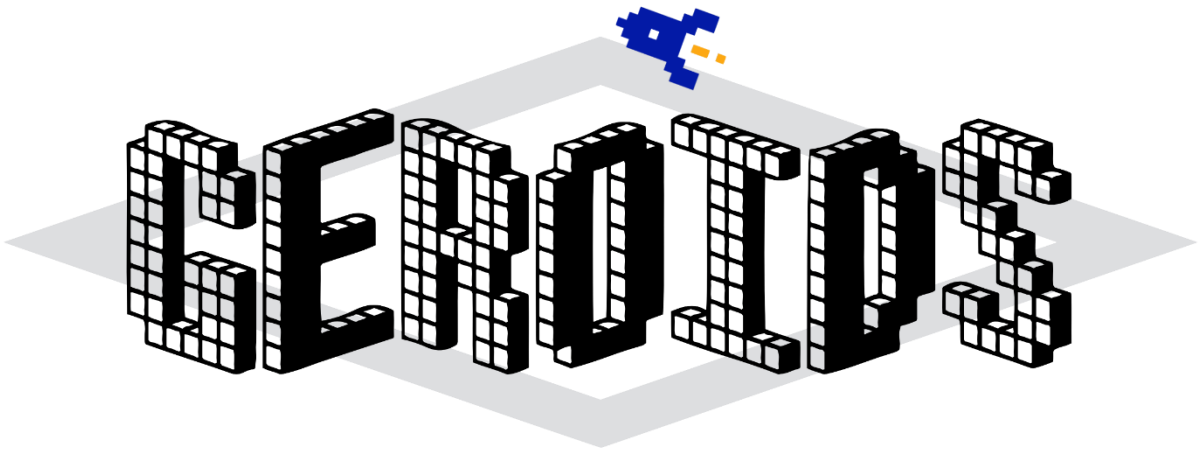
Anforderungsanalyse Geroids

EIN 2D-ARCADEGAME FÜR KLEIN UND GROSS



Projektteam PSIT3 Gruppe 01

Arben Shabani (PL)
Linda Bödi
Valentin Bossi
Matthias Kaderli

Inhaltsverzeichnis

1. Projektmanagement

Die Projektskizze ist fertig gestellt und die nächsten beiden Iterationsphasen geplant. Folgend sind die Planungen der Iterationen zwei und drei innerhalb der *Elaboration Phase* aufgeführt.

Iteration 2 [Elaboration Phase]

Task #	Arbeitspaket	Aufwand [h]	Ist [h]	Verantwortlich
1	Entwicklungsumgebung aufsetzen	10	7	Matthias K.
2	GUI-Entwurf fertig stellen	8	5	Linda B.
3	Domänenmodell erstellen	10	8	Arben S. (PL)
4	Analyse Architektur	10	8	Valentin B.
5	Statusmeeting	4	4	Alle
6	UC1 Registration beschrieben	4	4	Valentin B.
7	UC2 Spiel starten beschrieben	4	4	Linda B.
8	UC3 Spielen beschrieben	5	4	Arben S. (PL)
9	UC4 Highscores anzeigen beschrieben	5	4	Matthias K.
Total		60	48	

Iteration 3 [Elaboration Phase]

Task #	Arbeitspaket	Aufwand [h]	Ist [h]	Verantwortlich
1	SW-Architektur	8	6	Arben S.(PL)
2	Design-Klassendiagramm Entwurf	10	8	Arben S (PL), Matthias K.
3	DB-Entwurf	4	3	Matthias K.
4	Statusmeeting	4	4	Alle
5	Domänenmodell verbessert	4	3	Alle
6	Analyse-Resultat-Dokument	25	25	Alle
7	Erste Klassen in Java	4	2	Matthias K.
8	Erster GUI-Entwurf in HTML/CSS	6		Valentin B.
9	Erster "Spike" unserer Software	15		Alle
Total		80	51	

2. Anwendungsfälle

2.1 UC1 Registration

Umfang: *Geroids*-Spielapplikation

Ebene: Subfunktion

Primärakteur: Spieler

Stakeholder und Interessenten:

- **Spieler:** Möchte *Geroids* spielen
- **Unternehmen:** Möchte dem Spieler einen Account zuweisen, um In-App-Käufe zu ermöglichen und so Geldfluss zu generieren.

Vorbedingungen:

- Der Spieler hat eine Verbindung zum Internet und befindet sich auf der *Geroids* Webseite.

Nachbedingungen:

- Der Spieler kann ein Spiel spielen und In-App-Käufe tätigen.

Standardablauf:

1. Der Spieler gibt seinen Spielernamen und sein Passwort ein.
2. Das System überprüft die Login-Daten.
3. Das System informiert den Spieler über die erfolgreiche Anmeldung und leitet ihn zum Spiel weiter.

Erweiterungen/Alternative Abläufe:

1a) Neuer Spieler ohne Login

1. Der Spieler gibt einen freien Spielernamen, ein Passwort und eine E-Mail-Adresse ein.
   Das System überprüft die Login-Daten.

   (Der Spieler wiederholt Schritt 1, wenn er keinen freien Spielernamen, kein Passwort, keine E-Mail-Adresse eingibt.)
2. Das System informiert den Spieler über die erfolgreiche Anmeldung und leitet ihn zum Spiel weiter.

1b) Passwort zurücksetzen

1. Der Spieler will das Passwort zurücksetzen und wählt die entsprechende Option aus.
2. Der Spieler gibt seine E-Mail-Adresse ein.
3. Das System prüft die E-Mail-Adresse.

   (Der Spieler wiederholt Schritt 2, wenn er keine gültige E-Mail-Adresse eingegeben hat.)
4. Das System informiert den Spieler über den Versand einer E-Mail mit einem neuen Passwort an die angegebene E-Mail-Adresse.

2a) Inkorrekte Login Daten

1. Der Spieler gibt seinen Spielernamen und sein Passwort ein.
2. Das System überprüft die Login Daten.
3. Das System informiert den Spieler über die inkorrekten Login Daten.

Besondere Anforderungen:

- Verschlüsselte Datenübertragung mit HTTPS

Liste der Technik- und Datenvariationen:

- Die Registration kann mit den Browsern Safari, Firefox und Google Chrome durchgeführt werden.

Häufigkeit des Auftretens:

- Bei jedem neuen Spieler.

2.2 UC2 Tätigen von In-App-Käufen

Umfang: *Geroids*-Spielapplikation

Ebene: Subfunktion

Primärakteur: Spieler

Stakeholder und Interessenten:

- **Spieler:** Will eine sichere und schnelle Zahlung in einem Spiel tätigen. Dabei sollen keine Zahlungsfehler auftreten und die Preise klar sein. Ein Kauf soll bestätigt werden und anschliessend der aktuelle Stand der gekauften Erweiterungen angezeigt werden.
- **Unternehmen:** Will dem Spieler einen sicheren Ablauf der Zahlung mit einer kleinen Fehlerrate des Systems garantieren. Ausserdem will sie einen möglichst hohen Gewinn durch jeden Kauf.
- **Investor:** Will einen möglichst hohen Gewinn durch jeden Kauf.
- **Externes Bezahlungssystem:** Benötigt eine digitale Kaufanfrage im korrekten Format. Eine durchgeführte Zahlung soll richtig an die Applikation weitergeleitet werden, damit die Erweiterung als gekauft im Spiel angezeigt wird

Vorbedingungen:

- Der Spieler ist registriert und angemeldet.

Erfolgsgarantie/Nachbedingungen: Der Kauf ist gespeichert und die Liste mit den Spiel-Erweiterungen auf dem aktuellsten Stand. Die Quittung wurde erzeugt und an den Spieler ausgestellt. Das Geld ist überwiesen.

Standardablauf:

1. Der Spieler kommt über das Hauptmenü zu den In-App-Käufen.
2. Das System zeigt eine Liste mit allen zu kaufenden Erweiterungen an. In der Liste sind die schon gekauften Erweiterungen, die nicht mehr gekauft werden können, ausgegraut.
3. Der Spieler wählt eine Erweiterung, die er kaufen will, aus der Liste aus.
4. Das System leitet den Spieler an ein externes Zahlungssystem weiter, wo er die Zahlung für die Erweiterung tätigen kann.
5. Der Spieler zahlt und das externe Zahlungssystem verarbeitet die Zahlung und sendet eine Quittung.
6. Das System protokolliert den abgeschlossenen Kauf und bringt die Liste mit den Erweiterungen auf den neusten Stand.

   (*Der Spieler wiederholt Schritt 2 - 6, bis er nichts mehr kaufen möchte.)*
7. Der Spieler verlässt die In-App-Käufe mit den gekauften Erweiterungen

Erweiterungen/Alternative Abläufe:

*a) Jederzeit kann das System versagen:

Um Wiederherstellung und korrekte Zahlung zu unterstützen, müssen alle Transaktionen immer wieder gespeichert werden. Alle Zustände können wiederhergestellt werden, egal bei welchem Schritt des Szenarios das System versagt hat.

1a) Der Spieler startet das Spiel erneut, meldet sich an und fordert ein Wiederherstellen des vorherigen Zustands

1. Das System rekonstruiert den vorherigen Zustand.
    1.1 Das System kann den vorherigen Zustand nicht wiederherstellen:
        1.1.1 Das System zeigt dem Spieler einen Fehler an, erfasst den Fehler und kehrt zum Standard-Status zurück.

4-5a) Der Spieler will den Kauf abbrechen

1. Der Spieler bricht den Kauf ab und kehrt zu der Liste mit den Erweiterungen zurück

4-5b) Das System kann nicht mit dem externen Zahlungssystem kommunizieren:

1. Das System stellt erneut eine Verbindung zu dem Zahlungssystem her.
    1.1 Das System hat immer noch keine Verbindung zu dem Zahlungssystem.
    1.2 Das System zeigt einen Fehler an.
        1.2.1 Der Kauf wird abgebrochen und der Spieler gebeten, den Kauf später noch einmal zu versuchen

Besondere Anforderungen:

- Eine robuste Wiederherstellung, wenn das System abbricht.
- Eine schnelle Kommunikation zwischen dem System und dem externen Zahlungssystem (Anfragen und Antworten innerhalb von zehn Sekunden.)

Liste der Technik- und Datenvariationen:

3a) Der Spieler tätigt einen Kauf per Mausklick

5a) Die Rechnung wird per Mail an den Spieler geschickt

Offene Fragen:

- Was gibt es für externe Zahlungssysteme, die sich leicht an das System anbinden lassen?

2.3 UC3 Spielen

Umfang: *Geroids*-Spielapplikation

Ebene: Anwenderziel

Primärakteur: Spieler

Stakeholder und Interessenten:

- **Spieler:** Will ein gutes und einfaches Spiel spielen, ohne sich lange in Anleitungen einlesen zu müssen.
- **Unternehmen:** Will ein Spiel mit Suchtpotential, damit bei Spielern der Wille steigt In-App-Käufe zu tätigen. Möchte durch das simple Retro-Spiel viele Spieler anlocken und dadurch auch mehr Umsatz erzielen.
- **Programmierer/Hersteller:** Will, dass das Spiel fehlerfrei läuft und einfach erreichbar sowie ohne weitere Erklärungen spielbar ist.

Vorbedingungen:

- Der Spieler ist registriert und angemeldet.

Nachbedingungen:

- Nach jedem Spieldurchlauf wird das Spielergebnis mit dem zugehörigen "Nickname" des Spielers gespeichert, auch bei einer manuellen Beendigung des Spiels.

Standardablauf:

1. Der Spieler startet das Spiel über das Hauptmenü.
2. Ein neuer Spieldurchlauf wird generiert. Die Spielapplikation erstellt das Spielfeld mit der Spielfigur des Spielers und setzt die Score auf 0.
3. Sogenannte "Geroids", die Spielgegner, beginnen von oben herunterzufallen.
4. Der Spieler muss den Spielgegnern ausweichen oder sie mittels Schüssen eliminieren.
5. Die Schritte 3 und 4 werden solange wiederholt, bis die Spielfigur von einem “Geroid“ getroffen wird.
6. Der Spieldurchlauf wird beendet und das Spielergebnis wird in die Highscores-Liste aufgenommen.
7. Dem Spieler werden die "Game-Over-Seite" und seine Platzierung mit dem gerade erreichten Ergebnis angezeigt.

Erweiterungen/Alternative Abläufe:

2a) Fehler bei der Generierung des Spieldurchlaufs

1. Bei der Generierung des Spieldurchlaufs gibt es einen Fehler.
2. Der Spieler beendet das Spiel und startet es neu.
3. Der Spieler startet einen neuen Spieldurchlauf.

2b) Fehler bei der Kommunikation zwischen Server und Client

1. Die Kommunikation zwischen Server und Client funktioniert nicht einwandfrei.
2. Die Spielfigur lässt sich nicht steuern und die "Geroids" fliegen nicht so, wie sie sollen.
3. Der Spieler beendet das Spiel und startet es neu.
4. Der Spieler startet einen neuen Spieldurchlauf.

6a) Fehler bei der Abschliessung des Spieldurchlaufes

1. Bei der Abschliessung des Spieldurchlaufs gibt es einen Fehler.
2. Der Spieler beendet das Spiel und startet es neu.

7a) Fehler beim Anzeigen der Platzierung

1. Auf der *Game-Over-Seite* wird die Platzierung nicht angezeigt.
2. Der Spieler beendet das Spiel und startet es neu.

Offene Fragen:

- Wird das Spiel in Level/Stufen gegliedert oder gibt es alternative Modi.
- Werden zusätzliche durch Zufallsgeneratoren generierte Erweiterungen implementiert, wie z.B. zeitlich limitierte Schutzschilde?

2.4 UC4 Highscore anzeigen

Umfang: *Geroids*-Spielapplikation

Ebene: Subfunktion

Primärakteur: Spieler

Stakeholders und Interessenten:

- **Spieler:** Will seine Leistung mit einer früheren oder mit anderen Spielern vergleichen.
- **Unternehmen:** Kompetitiven Geist im Spieler wecken, um den Spieler zum Weiterspielen zu bringen und andere Spieler einzuladen. Will die Chance auf mehr Umsatz erhöhen.

Vorbedingungen:

- Der Spieler befindet sich im Hauptmenü oder hat ein Spiel verloren.

Standardablauf:

1. Der Spieler gelangt über das Hauptmenü bei den Highscores.
2. Das System lädt die Top zehn Einträge und präsentiert sie mit absteigender Punktezahl in einer Liste.

Erweiterungen oder alternative Abläufe:

1a) Der Spieler gelangt von einem beendeten Spiel bei den Highscores.

1. Der Spieler befindet sich im Spiel und verliert.
2. Der Spieler gelangt über das Hauptmenü bei den Highscores.
3. Das System lädt die Top zehn Einträge und präsentiert sie mit absteigender Punktezahl in einer Liste

2a) Das System kann keine Verbindung zur Datenbank mit den Punkteständen herstellen.

1. Der Spieler gelangt über das Hauptmenü bei den Highscores.
2. Das System kann keine Verbindung zur Datenbank mit den Punkteständen aufbauen und zeigt dem Spieler eine Fehlermeldung an.

Offene Fragen

- Soll der Spieler die Highscores Einträge sortieren können?
- Wie viele Spielergebnisse sollen in der Highscores-Liste angezeigt werden?

3. Anwendungsfalldiagramm

Folgend ist das Anwendungsfalldiagramm mit allen Anwendungsfällen aufgeführt. Das einzige externe System ist das Zahlungssystem, welches bei den In-App-Käufen zum Einsatz kommt. Der einzige Benutzer ist der Spieler; ein Administrator ist nicht vorgesehen.

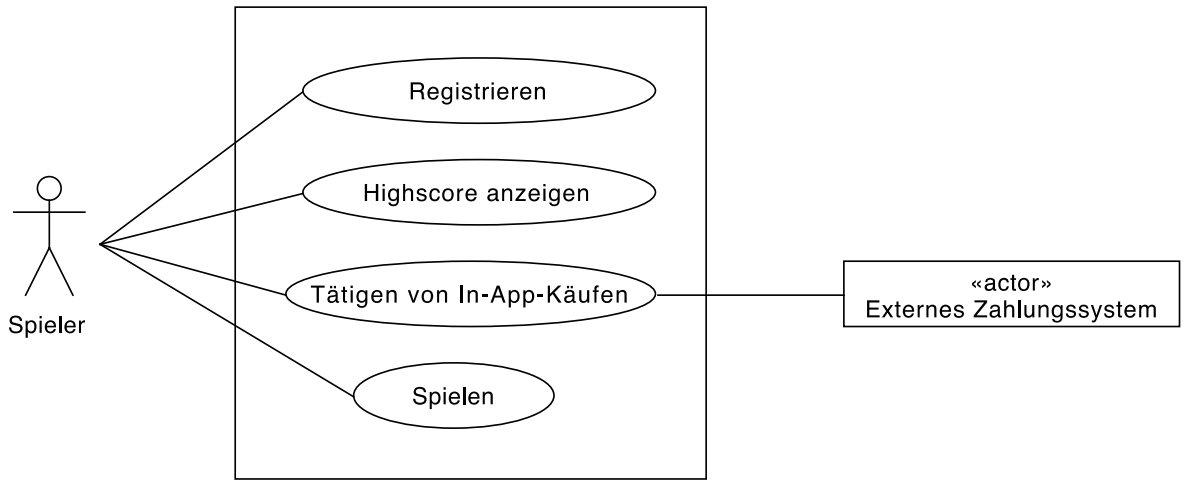


Abbildung 1 Anwendungsfalldiagramm

4. Domänenmodell

Nachstehend ist das Domänenmodell ersichtlich. Dieses dient als Grundlage für das Design-Klassendiagramm, das in einem späteren Schritt erstellt wird. Ausserdem dient das Domänenmodell zur Abgrenzung der einzelnen Domänen sowie für das allgemeine Verständnis der Zusammenhänge.

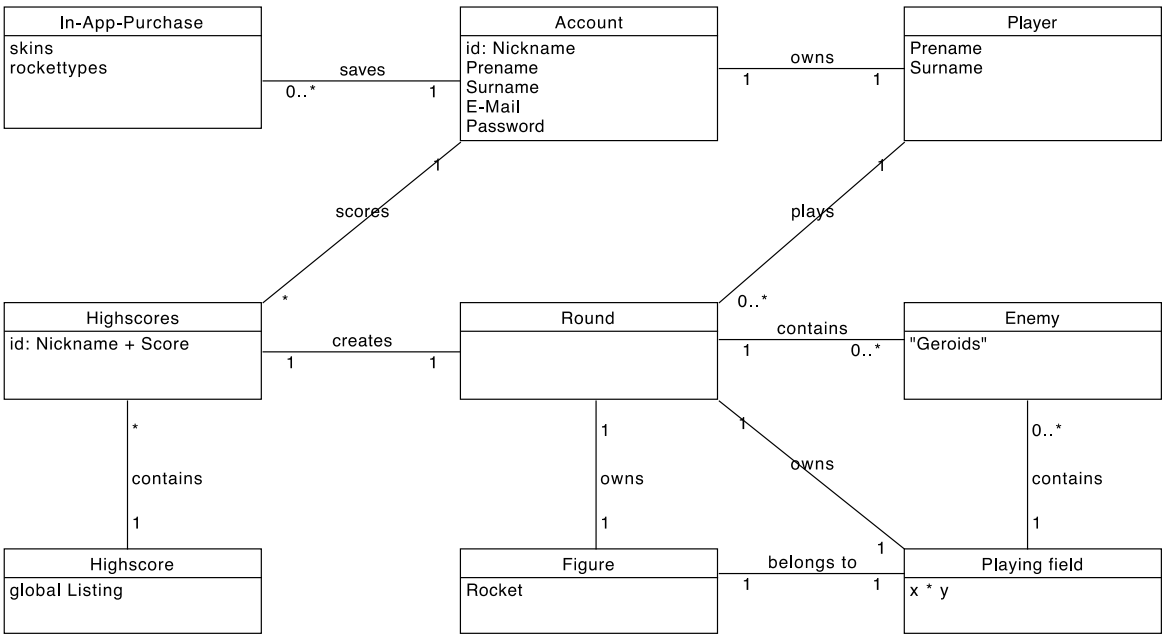


Abbildung 2 Domänenmodell

5. Eine erste Architektur

Im Bild unten ist die logische Architektur ersichtlich, welche die Abhängigkeiten und die Teil-Systeme der Applikation aufzeigt.

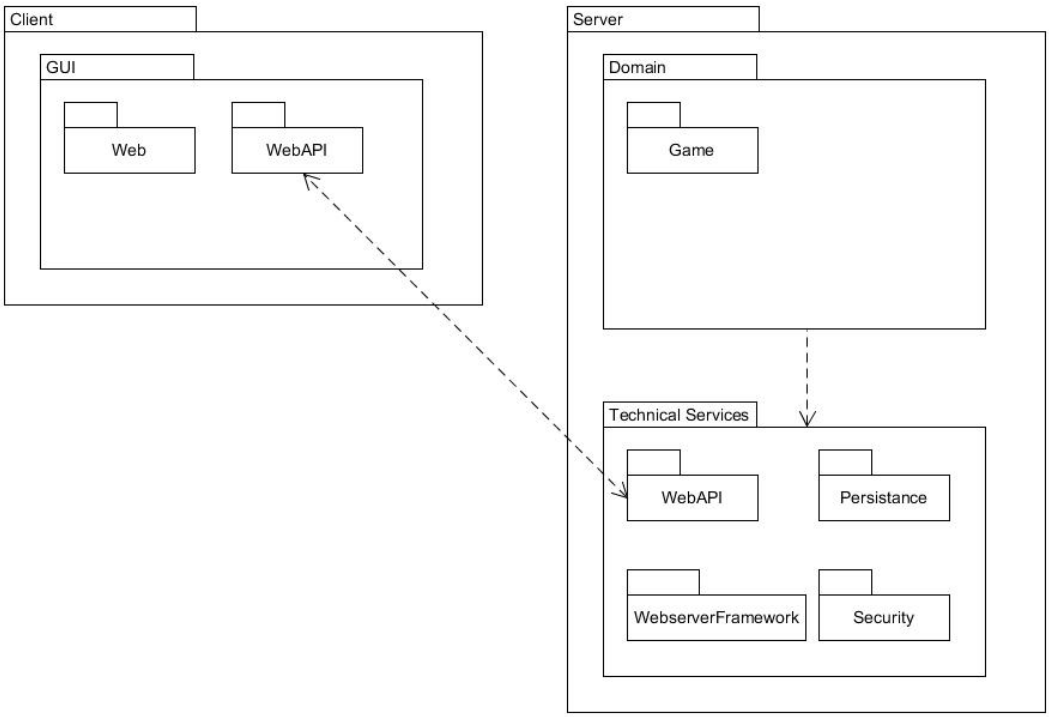


Abbildung 3 Logische Architektur

Das *GUI* ist die optische Darstellung, die mit HTML vor allem mit Canvas, CSS und JavaScript umgesetzt wird und bildet mit der WebAPI den *Client*. Auf der Serverseite haben wir die *Domain* mit dem Spiel und die *Technical Services* mit der Datenerhaltung, dem Webserverframework und der Sicherheitsschicht. Ebenfalls hier erhalten ist die WebAPI um mit dem Client zu kommunizieren.

In der untenstehenden Abbildung ist der erste Entwurf der Datenbank aufgeführt (ER-Diagramm).

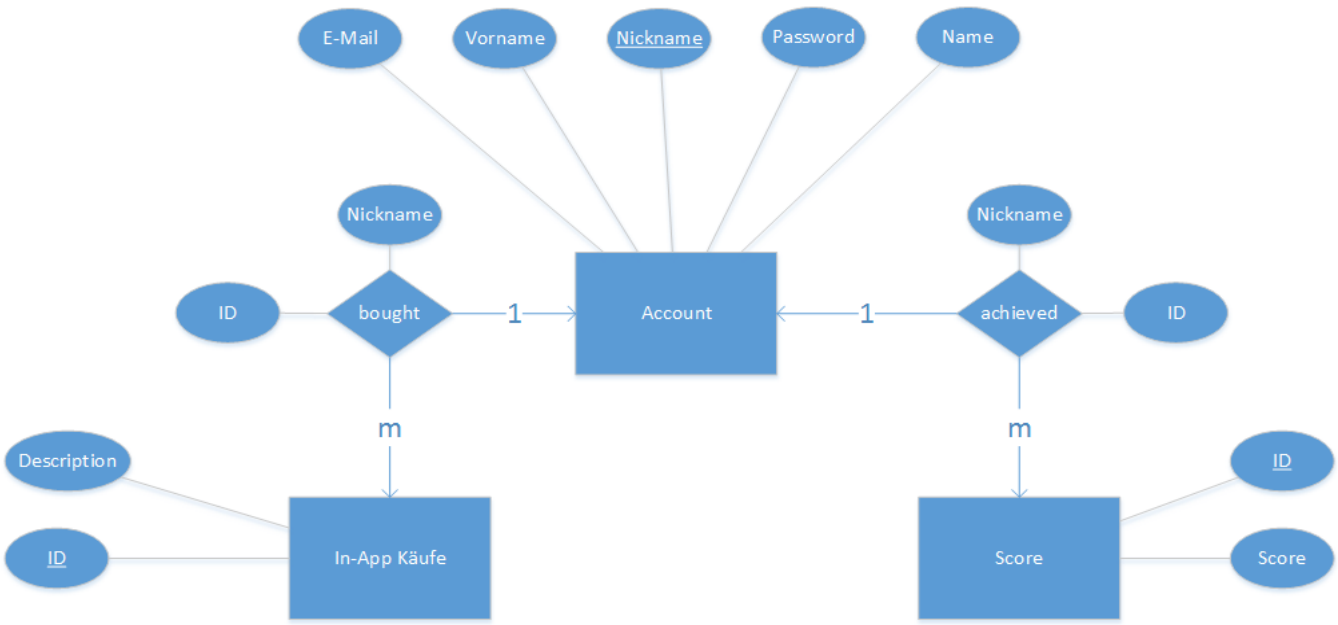


Abbildung 4 Entwurf DB Design

Die Datenbank wird simpel gehalten mit drei Tabellen und zwei Beziehungen. Die Tabelle Account ist über die Beziehung *bought* mit der Tabelle *In-App Käufe*, in der alle Erweiterungen, die gekauft werden können gespeichert sind, verbunden. Zusätzlich existiert mit *achieved* eine Beziehung zwischen der Tabelle *Account*, in der die Spielerkonten gespeichert werden, und der Tabelle *Score,* die alle Spielergebnisse speichert.

6. Zusätzliche Spezifikationen

Funktionalität

- Der Spielername wird in einer Datenbank gespeichert und ist einmalig.
- Der Benutzer registriert sich mit Name, Vorname, *Nickname*, Email und einem Passwort.
- In-App-Käufe ermöglichen es, das Raumschiff mit zusätzlichen Funktionalitäten auszustatten.
- Sicherheit: In-App-Käufe werden mit branchenüblichen Sicherheitsstandards geschützt.
- Mögliche Weiterentwicklungen der Applikation: Mehrspielermodus, auf Smartphones lauffähig, weitere Schwierigkeitsmodi.

Benutzerfreundlichkeit

- Die Applikation funktioniert in den aktuellen Versionen von Safari, Firefox, Google Chrome.

Verlässlichkeit

- Die Applikation soll 24 Stunden und Sieben Tage in der Woche für die Spieler verfügbar sein.

Leistung

- Die Applikation soll von 10’000 Spielern gleichzeitig benutzt werden können.
- Das Spielvergnügen soll von schlechter Netzwerkverbindung nicht beeinträchtigt werden, sondern immer flüssig laufen.

Wartung

- OO-Software-Design
- Spiellogik soll in Java programmiert sein.

Spielregeln / Domain Rules

- Die *Geroids*-Applikation startet im ersten Level. Das Raumschiff hat Standardschüsse und die "Geroids" eine Standardgeschwindigkeit, mit der sie in die Richtung des Raumschiffes fliegen.
- Im ersten Level braucht das Raumschiff einen Standardschuss, damit ein "Geroid" zerstört wird. Ein zerstörter "Geroid" gibt einen Punkt.
- Im ersten Level hat man einen Schutzschild mit einem Punkt. Wenn ein "Geroid" das Raumschiff trifft, verringert dies sein Schutzschild um einen Punkt. Bei null Punkten des Schutzschilds ist das Spiel beendet.
- In jedem weiteren Level erhöht sich die Geschwindigkeit der "Geroids".

7. System-Sequenzdiagramm (SSD)

Nachfolgend ist das System-Sequenzdiagramm für den Anwendungsfall *Registration* aufgeführt. Es ist gut ersichtlich wie und mit welchen Systemen kommuniziert wird.

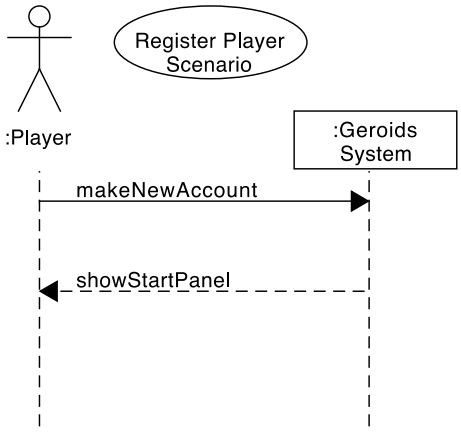


Abbildung 5 UC1 Spieler registrieren

Nachfolgend ist das System-Sequenzdiagramm für den Anwendungsfall *Tätigen von In-App-Käufen* aufgeführt.

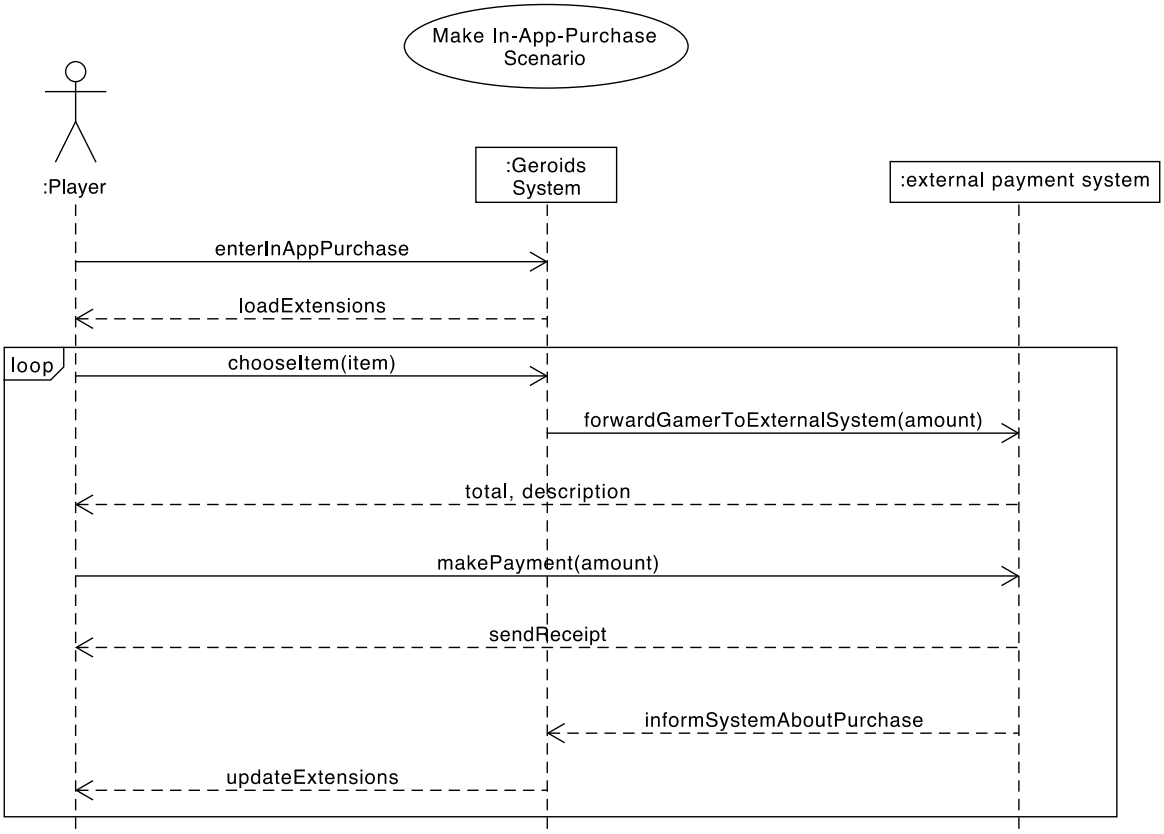


Abbildung 6 UC2 Tätigen von In-App-Käufen

Nachfolgend ist das System-Sequenzdiagramm für den Anwendungsfall *Spielen* aufgeführt.

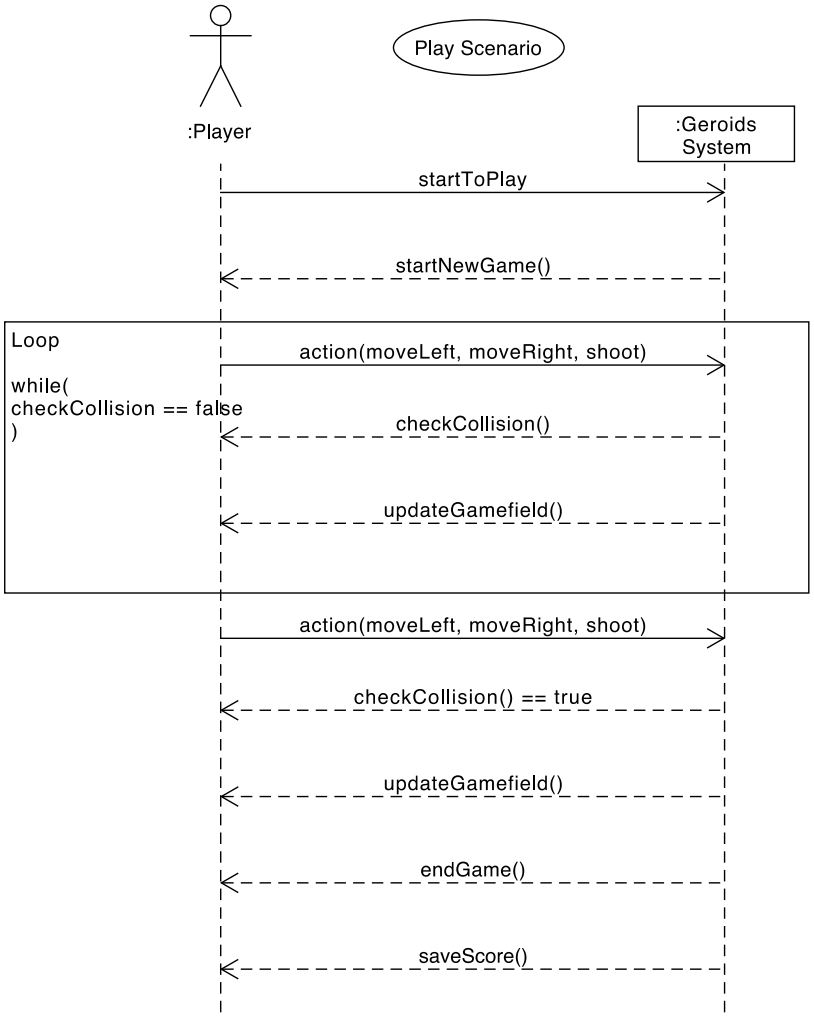


Abbildung 7 UC3 Spielen

Nachfolgend ist das System-Sequenzdiagramm für den Anwendungsfall *Highscore anzeigen* aufgeführt.

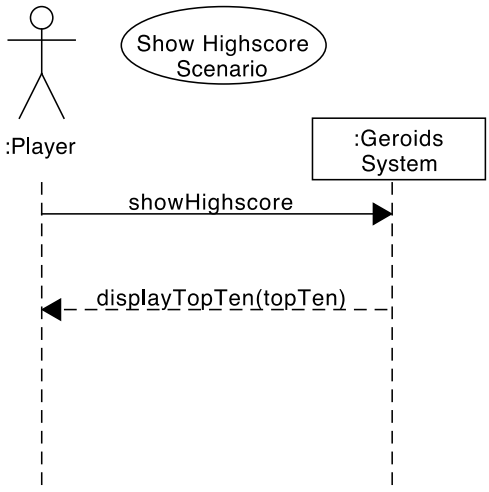


Abbildung 8 UC4 Highscore anzeigen

8. Systemoperationen

8.1 UC1 Registration

makeNewAccount
Operation: makeNewAccount()
Vorbedingung: Der Spieler hat noch kein Spielerkonto.
Nachbedingung: Mit der Operation makeNewAccount wird ein neues Spielerkonto im System erstellt. In die Datenbank werden Vorname, Nachname, *Nickname*, Passwort und E-Mail-Adresse eingetragen. Im System wird eine Spielerinstanz erstellt.

showStartPanel
Operation: showStartPanel()
Vorbedingung: Der Spieler ist registriert.
Nachbedingung: Das System zeigt dem Spieler mit showStartPanel die Startoberfläche des Spiels an.

8.2 UC2 Tätigen von In-App-Käufen

enterInAppPurchase
Operation: enterInAppPurchase()
Vorbedingung: Der Spieler ist registriert und eingeloggt.
Nachbedingung: Der Spieler sieht eine Liste mit Erweiterungen, die gekauft werden können.

chooseItem
Operation: chooseItem(item)
Vorbedingung: Der Spieler befindet sich bei den InAppKäufen.
Nachbedingung: Der Spieler hat die Erweiterung ausgewählt, die er kaufen möchte.

forwardGamerToExternalSystem
Operation: forwardGamerToExternalSystem()
Vorbedingung: Der Spieler hat eine Erweiterung gewählt, die er kaufen möchte.
Nachbedingung: Der Spieler wurde an das externe Zahlungssystem weitergeleitet, wo er die Zahlung tätigen kann.

makePayment
Operation: makePayment(amount)
Vorbedingung: Der Spieler möchte eine Erweiterung kaufen und befindet sich beim externen Zahlungssystem.
Nachbedingung: Der Spieler hat bezahlt, die Erweiterung ist im Spiel eingebunden, die Liste mit den Erweiterungen aktualisiert und es wurde ihm eine Quittung zugestellt.

8.3 UC3 Spielen

startNewGame
Operation: startNewGame()
Vorbedingung: Spieler befindet sich im Hauptmenü und drückt auf "Spiel starten".
Nachbedingung: Weiterleitung auf die "Spiel"-Seite und das System erstellt einen neuen Spieldurchlauf.

checkCollision
Operation: checkCollision()
Vorbedingung: Ein Spielzug wurde vollzogen, beispielsweise "nach rechts/links" bewegen oder "schiessen".
Nachbedingung: Solange keine Kollision stattgefunden hat, kann das Spiel weiterlaufen.

updateGamefield
Operation: updateGamefield()
Vorbedingung: Ein Spielzug wurde vollzogen, beispielsweise "nach rechts/links" bewegen oder "schiessen".
Nachbedingung: Das Spielfeld wurde aktualisiert und wird der GUI übergeben zur optischen Darstellung.

endGame
Operation: endGame()
Vorbedingung: Es fand eine Kollision zwischen einem Spielgegner und der Spielfigur statt.
Nachbedingung: Das Spiel wurde beendet und der "Game Over"-Banner erscheint.

saveScore
Operation: saveScore()
Vorbedingung: Das Spiel wurde beendet und der Score steht fest.
Nachbedingung: Der Score wird in der Datenbank abgespeichert und die Highscore-Liste aufgenommen.

8.4 UC4 Highscore anzeigen

pressHighScoreButton
Operation: pressHighScoreButton()
Vorbedingung: Der Spieler befindet sich im Hauptmenü
Nachbedingung: Weiterleitung auf eine neue Instanz der Highscores Seite

forewardToHighscorePage
Operation: forewardToHighscorePage()
Vorbedingung: Der Spieler hat eine Anfrage an das System gesendet für die Highscores Liste.
Nachbedingung: Spieler wird an eine Instanz der Highscores-Seite weitergeleitet. Eine Anfrage für die Punktestände wurde abgeschickt.

displayTopTen
Operation: forewardToHighscorePage(topTen)
Vorbedingung: Der Spieler hat eine Anfrage für die Highscores-Liste an das System gesendet.
Nachbedingung: Die Punktestände werden auf der neuen Instanz der Highscores-Seite angezeigt.

9. Glossar

Geroids: Arcadespiel als Webapplikation; Name aus der Verschmelzung der beiden Wörter Geometry und Asteroids, eines der ersten Arcadespiele. Die Spielapplikation *Geroids* wird immer kursiv geschrieben um es von den Spielgegnern, den "Geroids", zu unterscheiden.

"Geroids": Spielgegner; geometrische Form, die durch das All fliegen, immer in Anführungs- und Schlusszeichen geschrieben um es von der *Geroids*-Applikation zu unterscheiden.

HTTPS: HyperText Transfer Protocol Secure (sicheres Hypertext-Übertragungsprotokoll) ist ein sicheres Kommunikationsprotokoll im World Wide Web

Browser: Computerprogramm zur Darstellung von Webseiten im World Wide Web

In-App-Kauf: Käufe, die man innerhalb einer Applikation tätigen kann

Nickname: Spitzname

Score: Punktestand

Highscores: die höchsten Punktstände, die erreicht wurden

OO-Software-Design: Objektorientiere Software Programmierung

Instanz: Ein konkretes Objekt (in der Informatik)