XML Technology - WS17 - Memory - Technische Universität München (TUM)

Paul Dressel
B.Sc. Wirtschaftsinformatik
Technische Universität München (TUM)

OsmanAzizullah
B.Sc. Wirtschaftsinformatik
Technische Universität München (TUM)

Thuy TienHoang
B.Sc. Wirtschaftsinformatik
Technische Universität München (TUM)

Trong TienDao
B.Sc. Wirtschaftsinformatik
Technische Universität München (TUM)

Die Entwicklung von Memory in XML

Dieses DocBook beinhaltet eine kleine Dokumentation von dem Design, der Implementation und dem Testen des Spieles Memory. Darüber hinaus werden die verschiedenen Entwicklungsphasen, die wichtigsten Anforderungen, die Architektur, das Klassendiagramm sowie die Herausforderungen und Lösungen innerhalb des Teams beschrieben. Am Ende wird ein Spielablauf dargestellt und im Detail erklärt. Die Bilder von dem GUI veranschaulichen dabei die möglichen Vorgänge des Spieles, die in unterschiedlichen Phasen erreicht werden können.

Memory Projekt
Einführung-Das Bachelorpraktikum

Das Spiel Memory, welches im Folgenden hier dokumentiert und präsentiert wird, wird von den oben genannten Teammitgliedern entwickelt. Das Projekt fand im Wintersemester 2017/2018 innerhalb des Bachelorpraktikums "XML Technologie" mit Prof. Brüggemann-Klein an der Technische Universität München (TUM) statt.

Ein Merkmal dieses Bachelorpraktikums ist die Herausforderung eines gut funktionierenden Memory Spiels, welches die komplette Spannweite des XMLs und dessen Technologien verwendet, um das Spiel zu entwickeln. Daher war der Unterricht so organisiert, dass wir am Anfang einen "Einführungskurs" von der Prof. Brüggemann-Klein erhielten, in denen die Studenten die Vielzahl der wichtigsten Technologien der XML-Familie kennenlernten.

Danach begann der zweite Teil des Kurses und der Zeitplan wurde von den einzelnen Teams kontrolliert. Dabei war es Aufgabe sich innerhalb der Gruppe zu organisieren, die vordefinierten Regeln zu implementieren, umzusetzen und schließlich in die Grafik einzubinden, zu testen und die Arbeit zu dokumentieren. Verschiedene Übungsblätter, die wir wöchentlich erhalten haben, gelten als Vorbereitung für das implementierte Spiel. An diese Meilensteine konnten wir uns orientieren und Parallelen zu unserem Spiel ziehen.

Die dritte Phase des Projekts bestand schließlich darin, die geleistete Arbeit zu präsentieren und ein Demospiel der Memory-Version vorzustellen.

Design

In diesem Projekt gilt es folgende Regeln umzusetzen: "Die Projektaufgabe ist Memory. Mehrere Personen sollen in einem gemeinsamen Browser-Fenster spielen können. Der Spielstand soll gespeichert werden können, so dass man das Spiel zu einem späteren Zeitpunkt fortsetzen kann. Ihre Memory-Anwendung soll eine Lounge anbieten, in der offene Spiele ausgewählt werden können oder ein neues Spiel gestartet werden kann. In der Lounge sollen sich Spieler/innen registrieren können. Die Lounge soll eine Highscore-Liste für abgeschlossene Spiele anbieten."

Anzahl der Spieler

Für die Anzahl der Spieler, die an einem Tisch spielen können, haben wir beschlossen, die Anzahl auf sechs zu begrenzen. Es ist möglich, ein Spiel mit weniger als sechs Spielern zu spielen. Es ist jedoch erforderlich, dass mindestens zwei Spieler instanziiert sind, wenn ein neues Spiel gestartet wird. Für die Auswahl der Anzahl können die Spieler am Anfang des Spielstartes auswählen. Dies wird im Spiel vermerkt. Wir haben uns dabei für diese Spieleranzahl entschieden, da wir finden, dass Memory ein

Konkurrenzspiel ist. Deswegen beginnt die Anzahl der Spieler ab zwei. Aus Gründen der besseren Skalierbarkeit und besserer Übersichtlichkeit in einem einzigen Browser entschieden wir uns die Version mit höchstens sechs Spielern zu spielen.

Kartenanzahl

Für das gesamte Kartenspiel an einem Tisch entschieden wir uns für ein Spielfeld mit einmal sechzehn, vierundzwanzig oder zweiunddreißig Karten, welche von den Spielern am Anfang der Spielrunde ausgewählt werden. Die verschiedene Kartenanzahl bietet den Spielern je nach der Anzahl der Spieler die Schwierigkeit des Spieles zu variieren. Jede Schwierigkeitsstufe arbeitet mit verschiedenen Karten, sodass eine Abwechslungen innerhalb einer Schwierigkeitsstufe stattfindet.

Kartendesign

Jede Schwierigkeitsstufe hat ein unterschiedliches Kartendesign. Im Allgemeinen sind die Karten mit dem selben komplexen Motiv bedruckt, jedoch unterscheiden sie sich durch verschiedene knalligen Farben voneinander. Das Motiv der Karte besteht aus verschiedenen geometrischen Figuren. Wir haben uns für diese Variante des Designs entschieden, da wir alle der Meinung waren, dass das Merken der verschiedenen Farben eine größere Herausforderung darstellt und es deswegen einen höheren Spielerfolg zur Folge hat.

Die Muster und die Farben der Karten wurden in der XSLT Datei durch SVG designt.

Musterkarte aus dem 16 Kartendeck

Musterkarte aus dem 24 Kartendeck

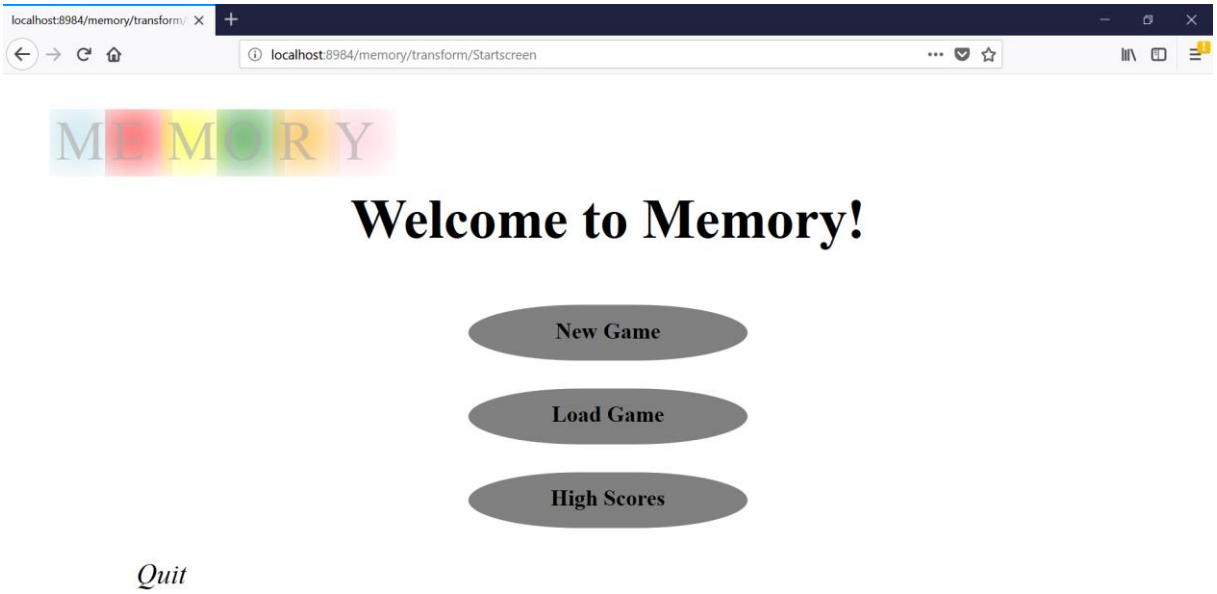
Musterkarte aus dem 32 Kartendeck

Ist die Karte noch nicht aufgedeckt, so ist die Rückseite der Karte weiß und ist mit dem Logo '' Memory '' gekennzeichnet.

Logo der umgedrehten Karte

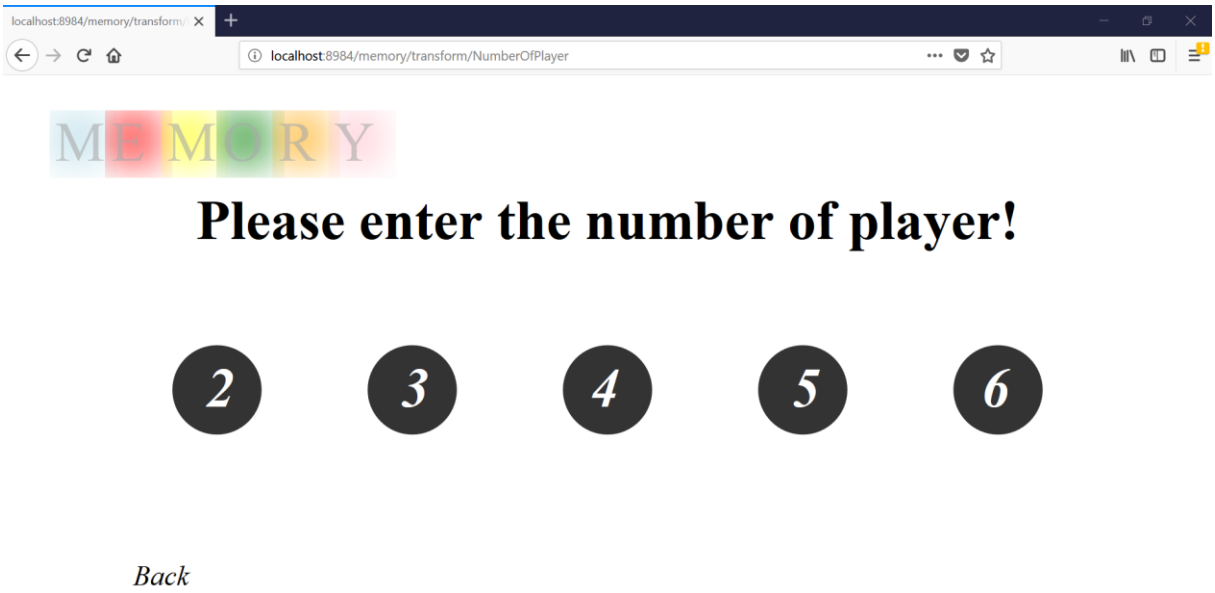
Einführung in das Spiel

Zu Beginn des Spiels befindet sich der Spieler in der Lounge, der im Screenshot drunter dargestellt wird. In der Lounge hat der Spieler die Möglichkeit, ein neues Spiel zu beginnen, ein begonnenes Spiel zu laden und sich den Highscore der abgeschlossenen Spiele anzuschauen. Falls der Spieler sich anders entscheidet und doch nicht spielen möchte, kann er auf den Quit-Button unten links drücken und das Fenster schließt sich.



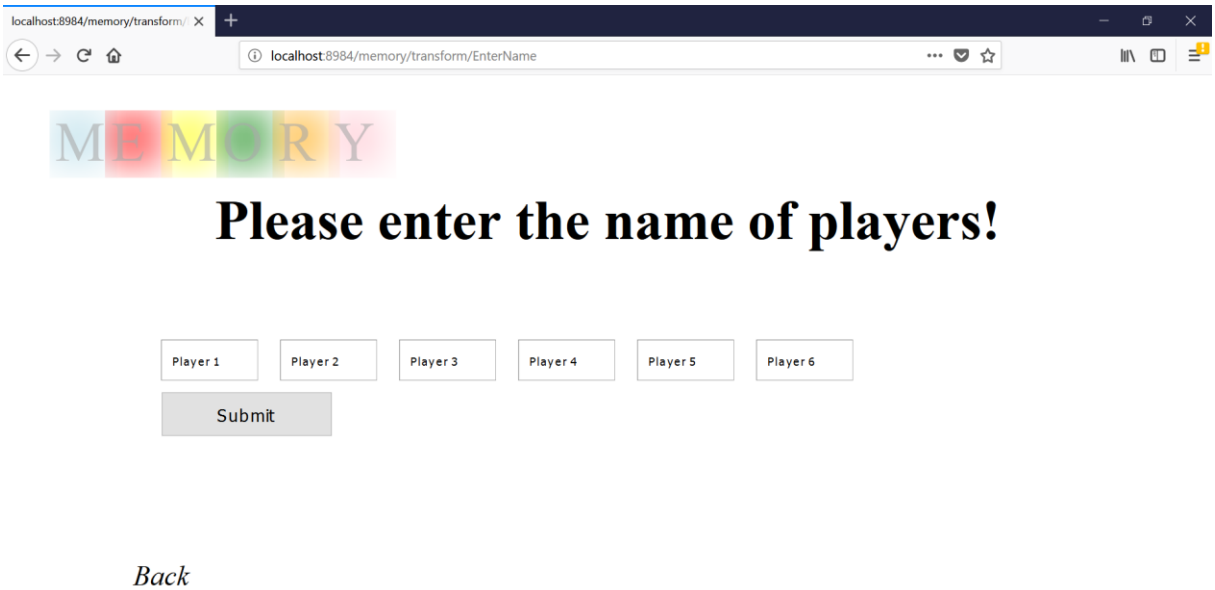
Lounge des Memoryspiels

Beginnt man ein neues Spiel, muss man sich zunächst für die Anzahl der Spieler entscheiden.



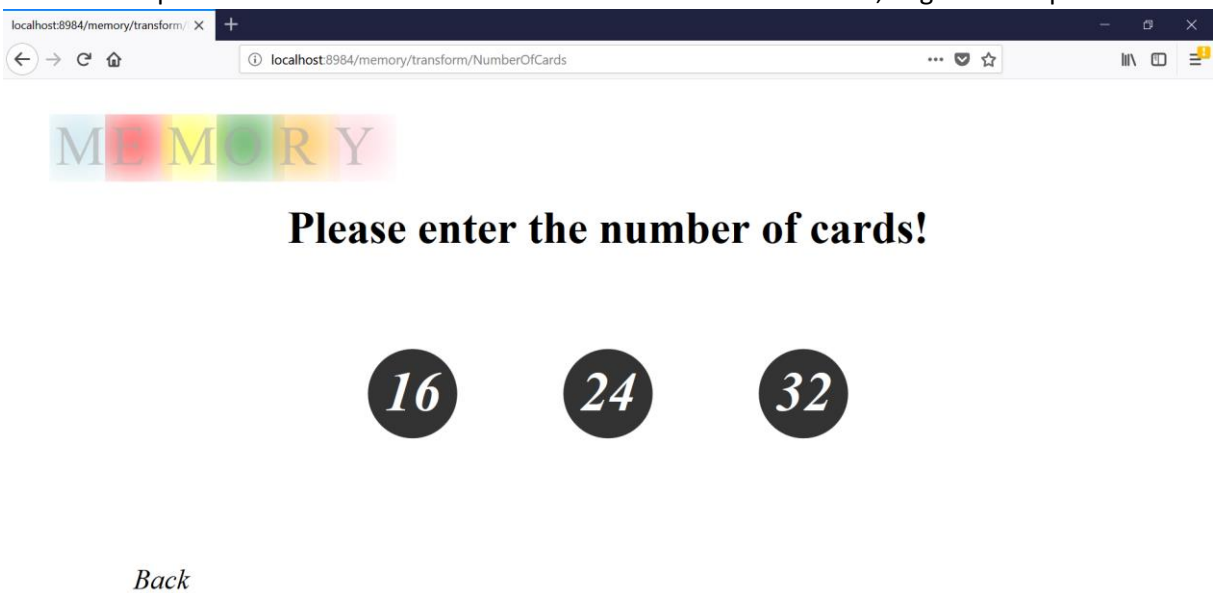
Auswahl der Spieleranzahl

Ist dies erledigt, müssen die Spieler nun ihren Namen eintragen.



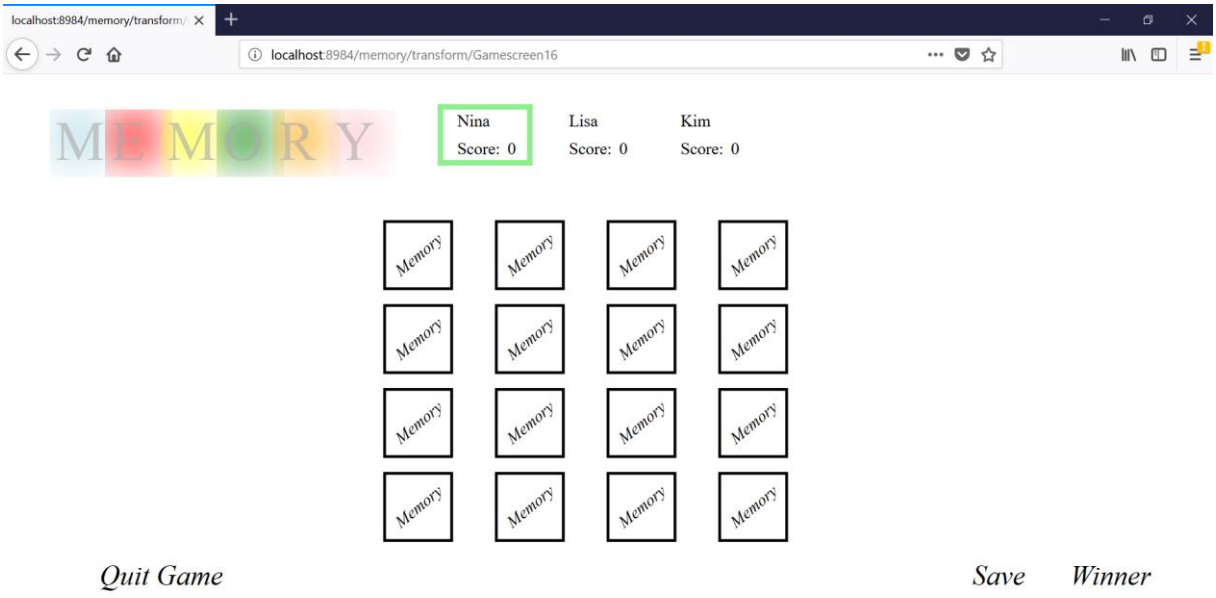
Eintragen der Spielernamen

Nachdem man sich für eine Anzahl an Spielern entschieden hat, wählt man die Anzahl der Karten mit denen man spielen möchte. Hat man sich für ein Kartendeck entschieden, beginnt das Spiel.

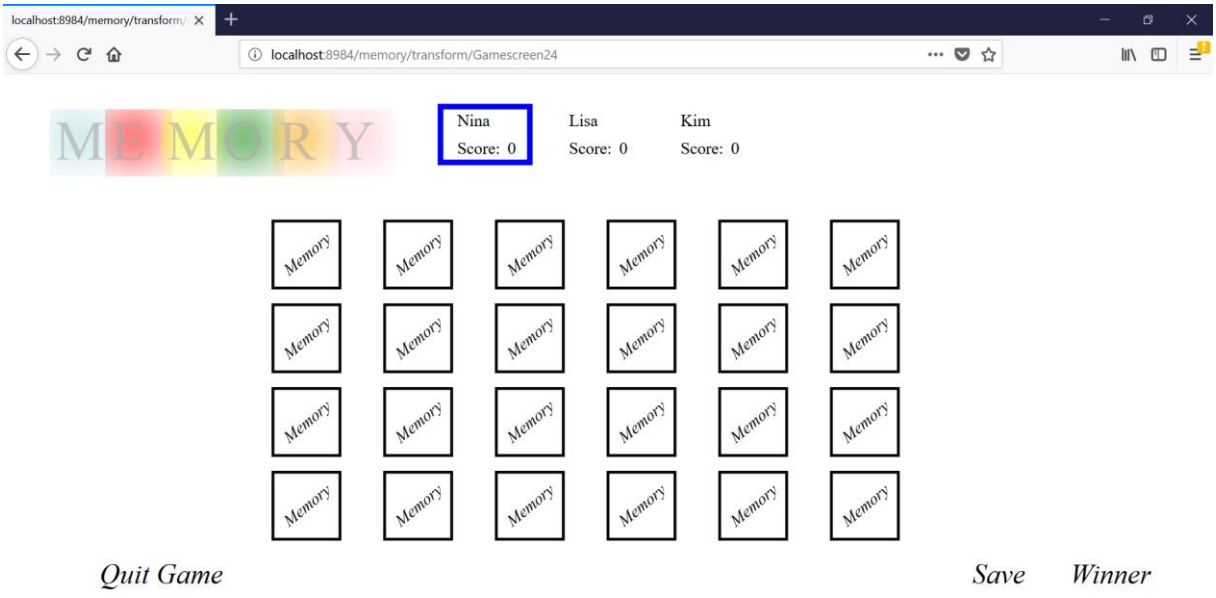


Auswahl des Kartendecks

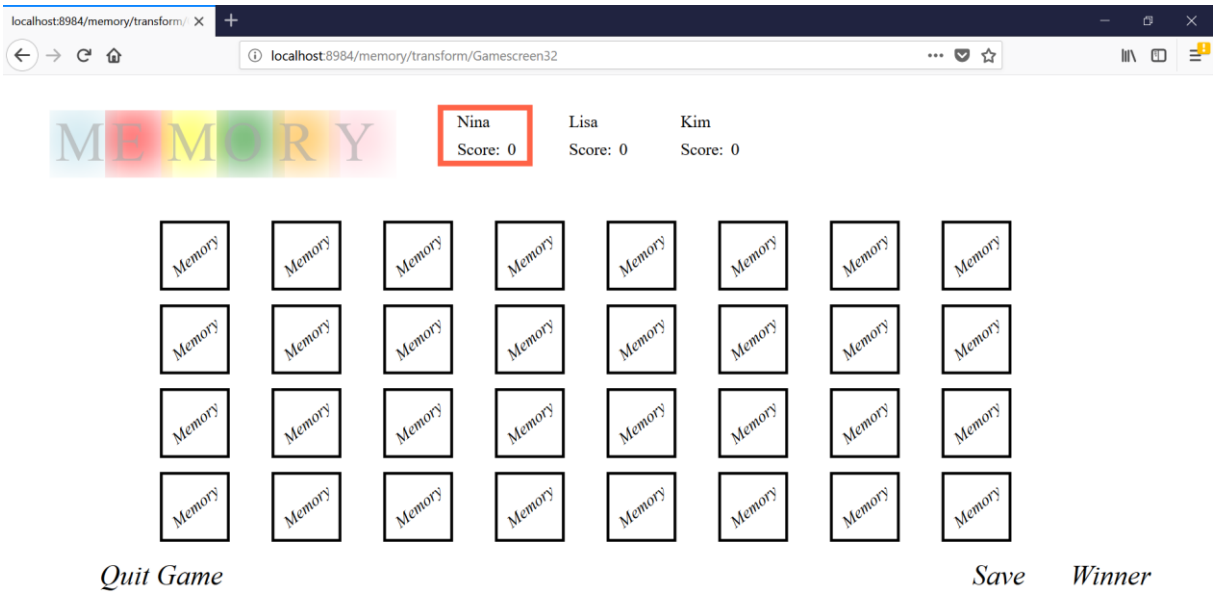
Ein laufendes Spiel kann man mit dem Quit-Button abbrechen. Danach kehrt der Spieler wieder zur Lounge zurück. Der aktuelle Spielstand wird hierbei nicht gespeichert.



Spiel mit drei Spielern und 16 Karten



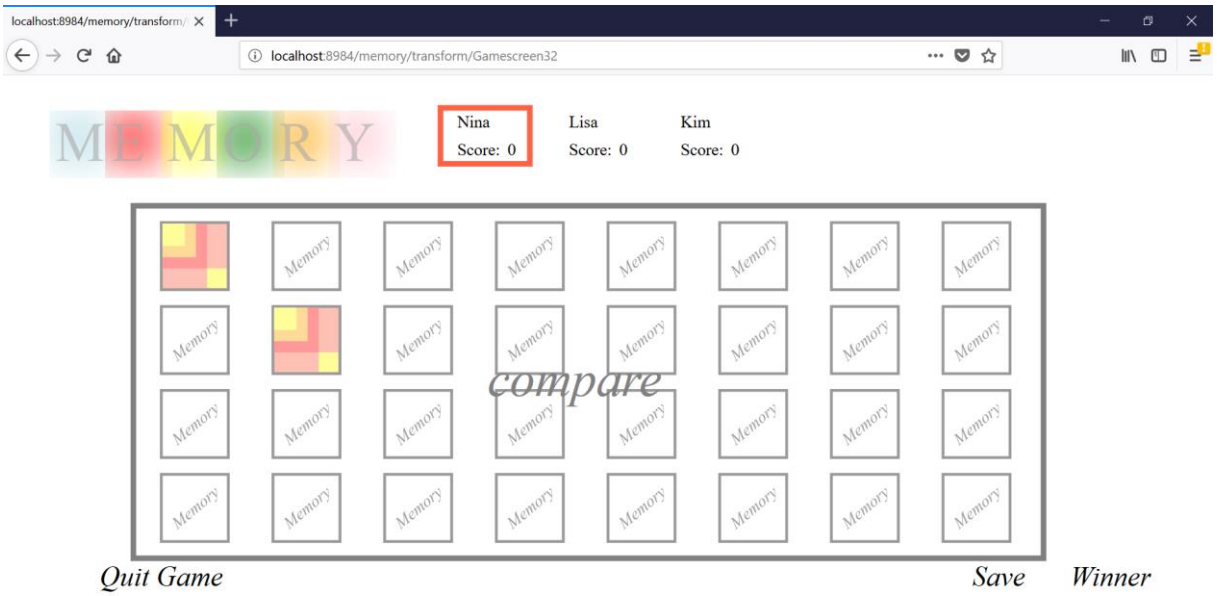
Spiel mit drei Spielern und 24 Karten



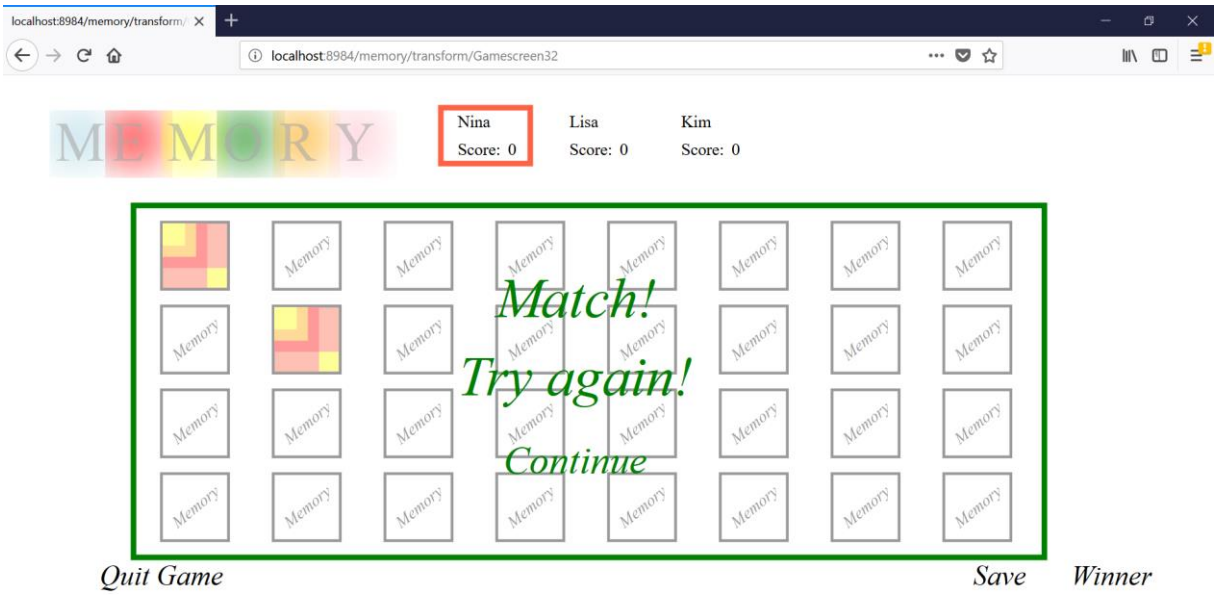
Spiel mit drei Spielern und 32 Karten

Spielablauf

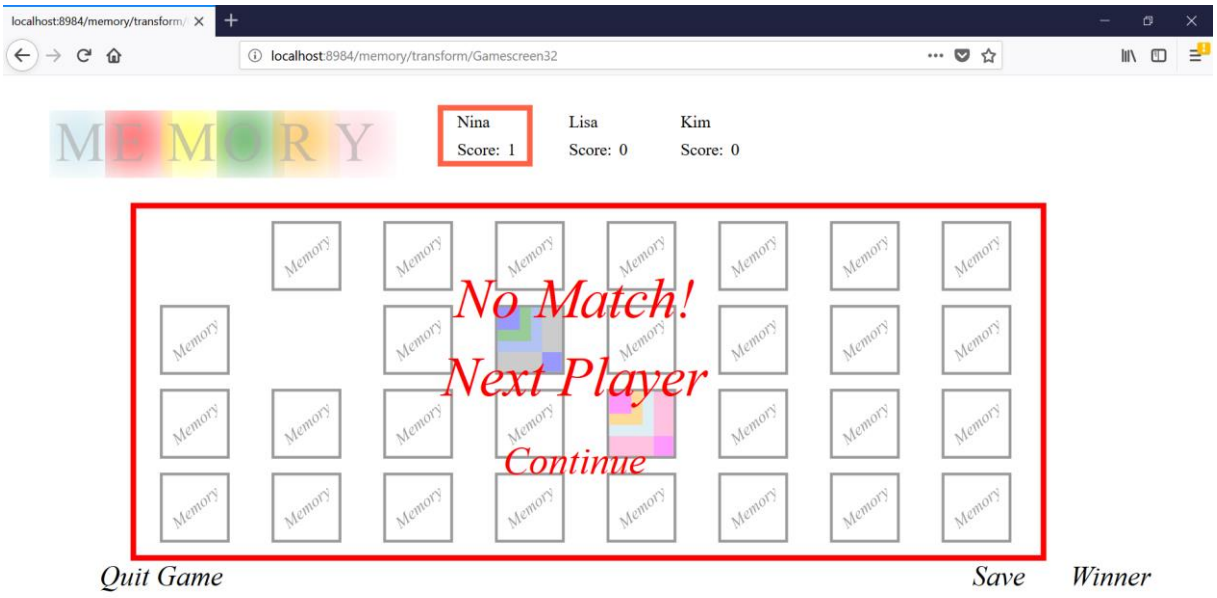
Das Spiel beginnt mit Player 1. Dieser hat die Möglichkeit, zwei verdeckte Karten aufzudecken. Bei Karten werden miteinander verglichen. Deckt er zwei identische Karten auf bekommt er einen Punkt, die Karten verschwinden und er darf weitere zwei Karten aufdecken. Deckt er zwei unidentische Karten auf gibt es keine Punkte und Player 2 ist an der Reihe.



Zwei Karten werden verglichen



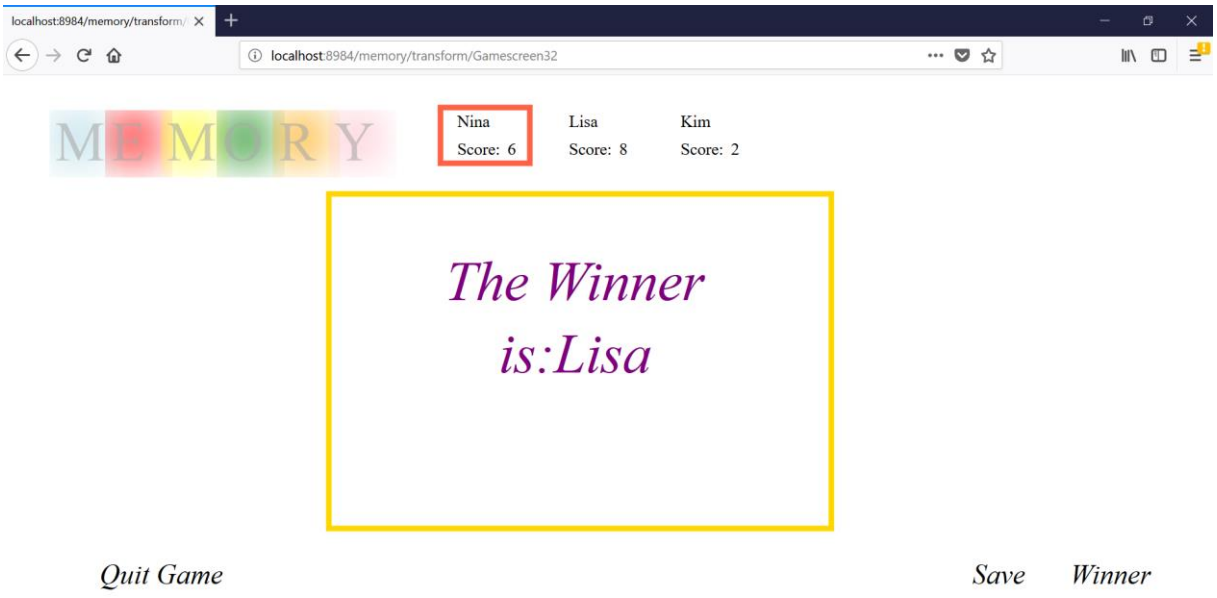
Zwei identische Karten wurden aufgedeckt



Zwei unidentische Karten wurden aufgedeckt

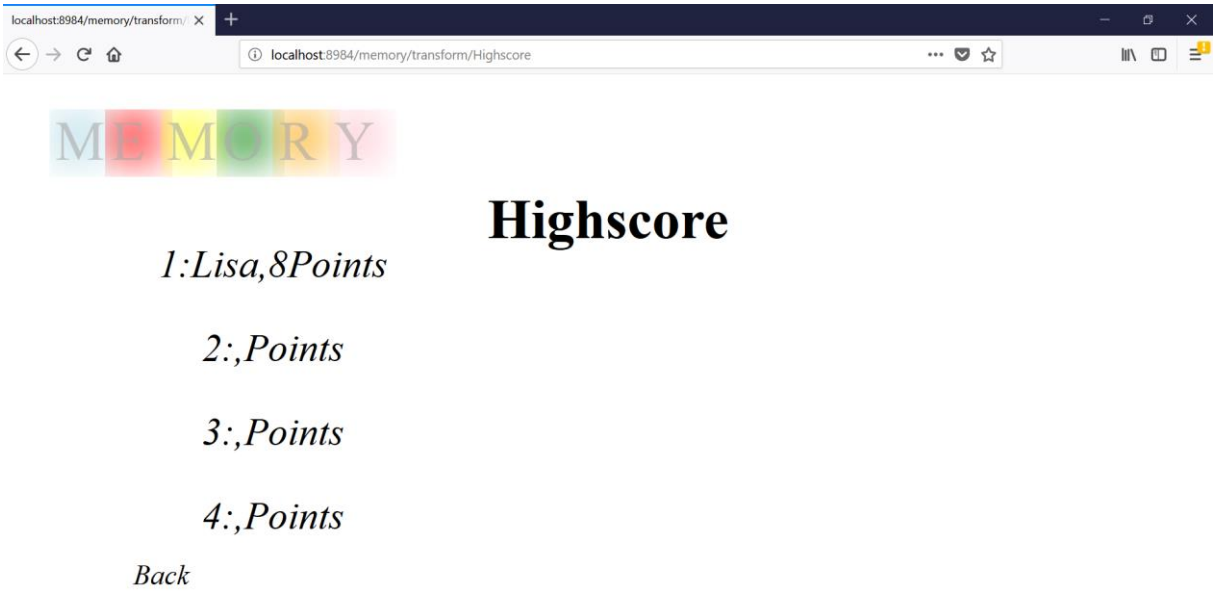
Gewinnstatus und Gewinner

Der Gewinnstatus jedes Spielers wird am Ende des Spiels überprüft. Das heißt, dass ein Spiel beendet ist sobald das letzte Memeorypaar gefunden wurde. Nachdem ein Spieler sich das letzte Paar rausgesucht hat, werden die Punkte aller Spieler miteinander verglichen. Der Spieler, der die meisten Memorypaare gefunden hat, hat in dieser Spielrunde gewonnen. Falls mindestens zwei Spieler dieselbe Punkteanzahl haben, so wird das Spiel als unentschieden betrachtet und das Spiel hat somit keinen Gewinner. Dabei klickt der Spieler auf den Winner-Button um den Gewinner anzeigen zu lassen.



Gewinnerausgabe einer Spielrunde

In der Lounge auf der Startseite finden die Spieler ein Highscore Tabelle. Der Gewinner des aktuellen Spieles wird mit dem Gewinner der vorherigen Spiele verglichen und nach absteigender Zahl der gefundenen Memorypaare sortiert und aufgelistet, sodass die Spieler ihre individuelle Bestleistungen sehen können.

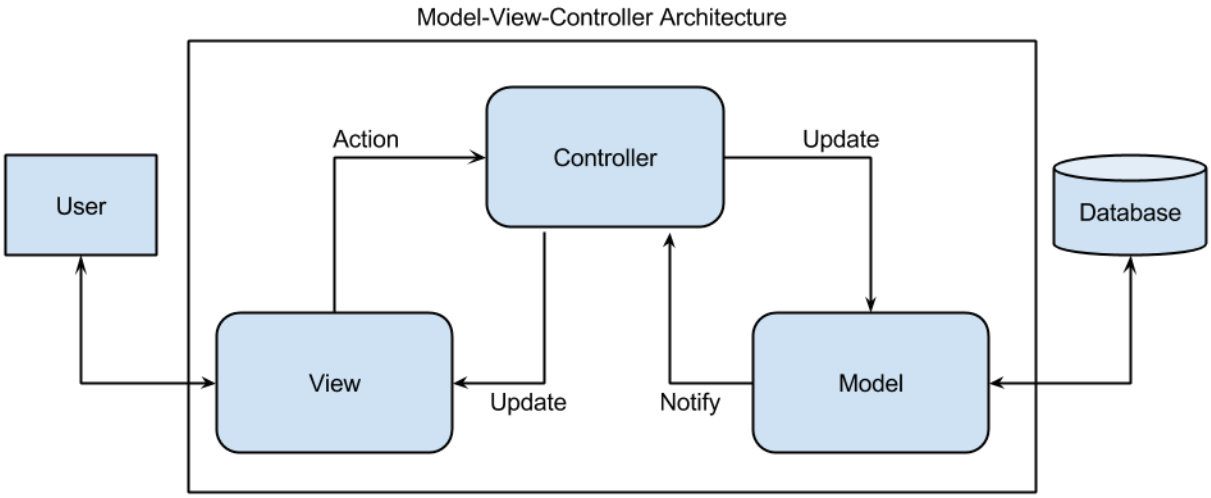


Highscoretabelle

Spiel speichern und laden

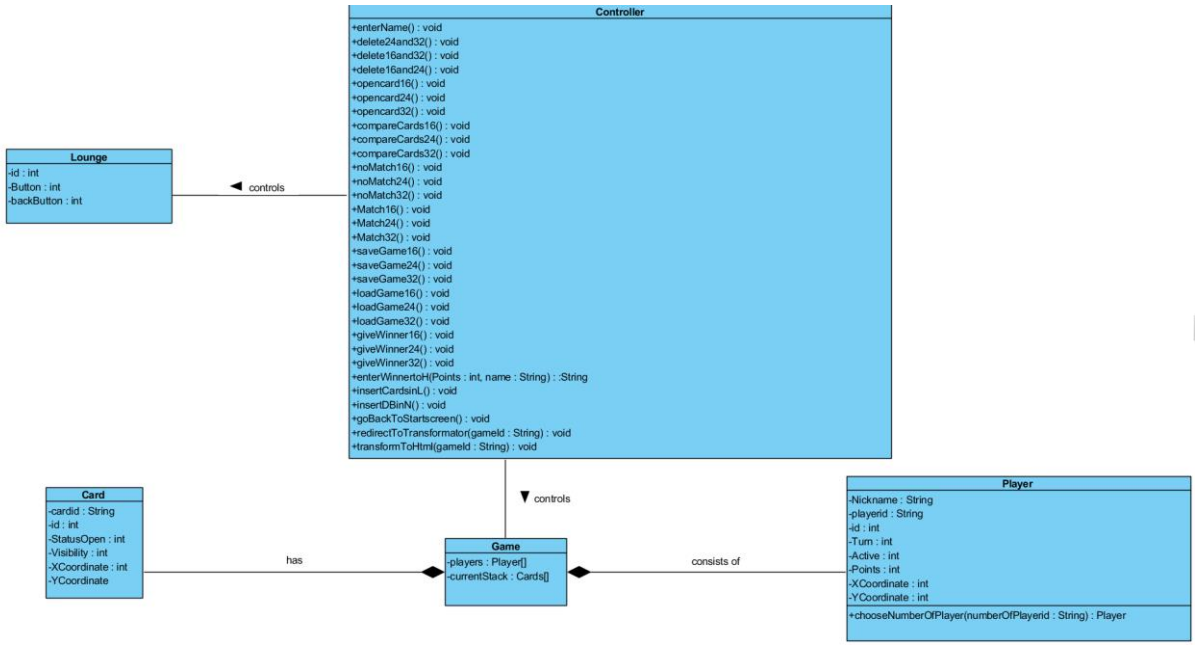
Die Spieler können ihren Spielstand zu jeder Zeit abspeichern und zu einem beliebigen Zeitpunkt das Spiel wieder fortführen. Das Laden von gespeicherten Spielen wird durch den Load-Game-Button in der Lounge ermöglicht. Danach hat der Spieler die Möglichkeit, ein gespeichertes Spiel mit entweder sechzehn, vierundzwanzig oder zweiunddreißig Karten fortzusetzen.

Implementierung
Architektur-MVC Modell



Die Implementierung unseres Memory-Spiels basiert auf einer Model-View-Controller-Architektur, welche im oben dargestellt ist. Das MVC-Modell beschreibt im Grunde genommen ein User Interface in Form von einem Model der realen Welt, welches durch eine View präsentiert wird, wobei Benutzeranfragen, wie z.B Mausklicks durch einen Controller gehandhabt werden. Der View ist die grafische Darstellung des Spiels, die Programmoberfläche (GUI), welche in der game.xsl implementiert ist. Der Controller fungiert als Vermittler zwischen der View und dem Model. Benutzeranfragen der Spieler leitet der View zum Controller, der die Spiellogik ausführt. Der Controller informiert der View über Änderungen am Model. Außerdem werden Änderungen am Model in der Datenbank gespeichert.

UML Klassendiagramm
Klassendiagramm des Memoryspiels



Im Folgenden werden die wichtigsten Funktionen des Memoryspiels erklärt, welche mit XQuery implementiert wurden

InsertCardsInDB.xqm:
Diese Funktion mischt die Karten vor jeder neuen Spielrunde neu, damit man nicht jede Spielrunde dieselbe Reihenfolge von Karten hat. Mit der Funktion random-number-generator werden die IDs der Karten (1-16, 1-24, 1-32) zufällig generiert. Die Kombination mit dem ?permute-Ausdruck gibt eine Permutation der zufälligen Zahlen zwischen 1-16, 1-24 bzw. 1-32 wieder. Die x- und y-Koordinaten der Karten werden ebenfalls in einer Liste gespeichert. Danach wird durch die Liste der IDs iteriert und nach jedem Durchlauf wird der ID die x- und y-Koordinate am jeweiligen Index i zugeordnet.

playersettings.xqm:
Hier wird die gewünschte Anzahl der Spieler, mit der man agieren möchte. Nachdem man sich für eine bestimmte Anzahl an Spielern entschieden hat, wird der Active-Status der Spieler in der Datenbank auf 1, also auf true gesetzt. Entscheidet man sich also für vier Spieler, wird der Active-Status der Spieler mit den IDs 1-4 folgerichtig auf 1 gesetzt. Ansonsten bleibt der Active-Status bei 0.

controller.xqm:
Im Controller wird die ganze Logik des Spieles implementiert. Um eine besser Übersicht über die Funktionen zu erhalten, wurden die meisten Funktionen in der turnCards.xqm und in der saveGame.xqm Datei implementiert und in der Controller Datei aufgerufen:

- enterName(): Die Namen der bestimmten Spieler werden von den Spielern eingegeben und eingelesen
- delete24and32(): Wird ein neues Spiel gestartet, erstellt er zunächst drei Datenbanken für die Fälle, ob man mit sechzehn, vierundzwanzig oder zweiunddreißig Karten spielen möchte. Nachdem man sich dann für die Anzahl der Spieler entschieden, hat wählt man dann die das gewünschte Kartendeck. Ist dies erledigt, wird das Spiel gestartet und die Datenbanken der Kartendecks, für die man sich nicht entschieden hat gelöscht.
- openCard16: Die Funktion setzt den StatusOpen der Karten auf 1, sodass die Karten aufgedeckt werden. Dabei wird geachtet, dass nur höchstens 2 Karten ausgewählt werden und nicht mehr.
- compareCards16(): Durch den Vergleich der IDs der Karten werden die Pärchen ermittelt. Bei jedem Spielzug wird der Spieler mit einem Popup informiert wie sein Spielzug verlaufen ist.
- noMatch16(): Wird kein Memorypaar gefunden, wird StatusOpen wieder auf 0 gesetzt und die Karten drehen sich um.
- Match16(): Wird ein Memorypaar gefunden, verschwinden die Karten, indem die Visibility auf 0 gesetzt wird.
- saveGame16(): Diese Funktion speichert den Spielstand des aktuellen Spieles. Dabei wird die Datenbank in eine andere Datei überschrieben
- loadGame(): Die Funktion nimmt die Daten vom gespeicherten Spiel und überschreibt in Gamescreen
- giveWinner16(): Der Gewinner des Spieles wird ermittelt. Dabei werden die Punkte der Spieler miteinander verglichen. Wer die höchste Punktzahl erreicht hat, gewinnt das Spiel. Haben mindestens zwei Spieler dieselbe hohe Punktzahl, wird dies als unentschieden ausgewertet.
- enterWinnerToH(Points:int, name:String) : Diese Funktion listet den Gewinner des aktuellen Spieles in die Highscoreliste und vergleicht ihn gleichzeitig mit den älteren Gewinner und sortiert diesen nach der Reihenfolge der Punkte.
- insertDBinL(): Klickt der Spieler auf den Load-Gam-Button, so wird die Datenbank des gespeicherten Spieles geladen und der Spieler kann sein Spiel fortsetzen.
- insertDBinN(): Dem Spieler wird ein neues Spielfeld errichtet, wenn er auf den New-Game-Button klickt.

- goBackToStartscreen(): Klickt der Spieler auf das Back-Button, wird er zurück zur Startseite geführt. Dabei werden alle Daten aus der Datenbank herausgelöscht, sodass der Spieler die Möglichkeit hat seine Eingaben neu zu tätigen.
- redirectToTransformator(gameId: String): Die Seite wird immer wieder neu geladen, wenn der Spieler auf einen bestimmten Button zwei Mal klickt.
- transformToHtml(gameId:String): Die Funktion nimmt die XML und XSLT Datei und transformiert die für den Webserver

Alle Funktionen, die für das Spielfeld mit 16 Karten beschrieben wurden, gilt analog auch für das Spielfeld mit 24 bzw. 32 Karten.

Testing

Umfassende Tests unserer Webanwendung erfolgten durch die Simulation von Anfragen mit speziell definierten Parametern und Funktionen. Die Art, wie sie verwendet werden, besteht darin, nur Anforderungen in der URL-Befehlszeile Ihres Browsers zu senden und zu sehen, was passiert. Effekte können dann in der baseX-Datenbank gesehen und mit dem angenommenen Ergebnis verglichen werden. Diese Anfragen werden dann an den Server und die Datenbank gesendet, wo die jeweiligen Ergebnisse in den XML-Daten beobachtet werden können. Bei einigen Grenzfällen wurde besonderer Fokus auf Tests gelegt. Auf diese Weise werden beispielsweise Integerüberläufe sicher gehandhabt. Darüber hinaus verlangte die Spiellogik manchmal, dass wir Sonderfälle sicher und realistisch gleichzeitig behandeln. Durch wiederholtes Testen dieser Fälle könnte die Wahrscheinlichkeit einer Fehleranfälligkeit effizient auf ein akzeptables Minimum reduziert werden.

Diese Vorgehensweise wurde durch die Unterscheidung zwischen Unit- und Integrationstests verstärkt, die natürlich beide wie oben beschrieben angewendet wurden.

Als ein weiterer sehr wichtiger Aspekt, der während der testgetriebenen Entwicklungszyklen berücksichtigt wurde, ist es wichtig zu erwähnen, dass wir als Team den geschriebenen Code des jeweils anderen durch Reviews doppelt und dreifach überprüft haben. Dies war sehr hilfreich in mehreren Aspekten. Auf diese Weise haben alle Teammitglieder die Logik aller anderen Komponenten der Anwendung verstanden. Zweitens haben wir viel voneinander gelernt, indem wir den Implementierungsstil und die Logik neu durchdacht haben. Und drittens konnten wir die hohe Qualität unserer Anwendung und ihres Codes sicherstellen, indem wir einige Fehler beseitigten, die sonst wahrscheinlich nicht gefunden worden wären.

Organisation
IDE

Die Entwicklungsumgbung für das Projekt war der Oxygen XML Editor, welcher von Prof.Brüggemann-Klein vorgeschlagen wurde. Die IDE unterstützt verschiedene Formate von Dateien und Programmiersprachen wie XSLT und XQuery. Für die Teammitglieder war es sehr hilfreich mit dieser IDE zu arbeiten. Dadurch war ein schneller Austausch von Informationen zwischen den einzelnen Mitgliedern möglich. Um den Beteiligten eine asynchrone Entwicklung zu ermöglichen, haben wir ein Repository auf GitHub erstellt. Diese Methode war sehr effizient, da alle Teammitglieder zu jeder Zeit auf dem aktuellen Entwicklungsstand waren. Außerdem konnten alle Teammitglieder unabhängig voneinander an ihrem eigenen Code arbeiten und anschließend in das Repository committen.

Management und Entwicklungsphase

Während des gesamten Projekts kümmerte sich Prof. Brüggemann-Klein um alle beteiligten Teams. Bei Fragen hinsichtlich der Organisation (Präsentationtermin und Abgabe) sowie Verständnis konnten wir uns jeder Zeit an Prof.Brüggemann-Klein wenden, die uns auch immer mit sehr hilfreichen Tipps weiterhelfen konnte, was von den Teammitgliedern sehr geschätzt wird. Da alle Teammitglieder sehr interssiert und begeistert von dem Projekt waren, gab es keine Probleme mit der Arbeitsmoral. Dies ermöglichte, dass wir uns schnell auf die Entwicklung des Memory-Spiels konzentrieren konnten und nicht unnötig viel Zeit in sowohl Planung als auch Diskussion steckten. Das einzige Problem hierbei war, dass drei Teammitglieder einen MacBook besitzen und das Spiel auf MacOs nicht gestartet bzw. gespielt werden konnte, weshalb hier das Installieren von Windows nötig war. Zu Beginn der Entwicklungsphase haben wir uns zu wöchtenlichen Scrum-meetings getroffen, um unseren aktuellen Stand, Fragen und Probleme sowie weitere Ideen zu besprechen. Während der sowohl Einzelnen als auch Teamarbeit am Projekt, haben sich die einzelnen Teammitglieder untereinander nochmal getroffen, um an Lösungen zu arbeiten. Die Ergebnisse wurden vor jedem Scrum-meeting auf Git veröffentlicht, damit jedes Teammitglied beim nächsten Treffen auf dem aktuellen Stand ist.

Schluss & Reflexion

Zusammenfassend kann man sagen, dass unsere entwickelte Memory-Anwendung die vordefinierten Anforderungen erfüllt. Natürlich kann man Erweiterungen vornehmen, wie z.B das Spielen über mehreren Geräten. Dies war jedoch nicht gefordert. Als Team haben wir auf einen hohen Wert auf sowohl Spiellogik als auch GUI gesetzt, um das Spielerlebnis der Spieler zu bereichern. Zum Abschluss kann man sagen, dass das Praktikum ein sehr interessanter und informativer Kurs war. Die Entwicklung einer Web-Anwedung ausschließlich mit Hilfe von XML-Technologien ist definitiv eine sehr spannende Sache, die wir als Team mit sehr viel Freude und Enthusiasmus angegangen sind. Was den Kurs noch aufregender gemacht hat war, dass wir nicht nur eine Anwendung entwickelt haben, sondern ein bekanntes Gemeinschafsspiel wie Memory. An dieser Stelle möchten wir uns als Team nochmal bei Prof. Brüggemann-Klein für die tolle Unterstützung bei Problemen sowie Ihrer Betreuung bedanken. Für weitere Interessierte ist dieser Kurs sehr empfehlenswert.