Der TitelPflichtenheft und technische Spezifikation im   
Programmierprojekt

**Projektname:** PuzzleMaster

**Mitarbeiter:** Dominik Probst, Patrick Giese, Justus Saringer

Inhaltsverzeichnis

[1 Visionen und Ziele 1](#_Toc56682265)

[2 Anforderungen an Ihr System 1](#_Toc56682266)

[2.1 Use-Cases 1](#_Toc56682267)

[2.2 Risiken 2](#_Toc56682268)

[2.3 GUI 3](#_Toc56682269)

[3 Realisierung 5](#_Toc56682270)

[3.1 Allgemeines 5](#_Toc56682271)

[3.2 Interne Schnittstellen 6](#_Toc56682272)

[3.3 Visual-Studio-Projektsetup 9](#_Toc56682273)

[3.4 Externe Schnittstellen 9](#_Toc56682274)

[4 Test und Implementierungsphase 10](#_Toc56682275)

[5 Lizenz 10](#_Toc56682276)

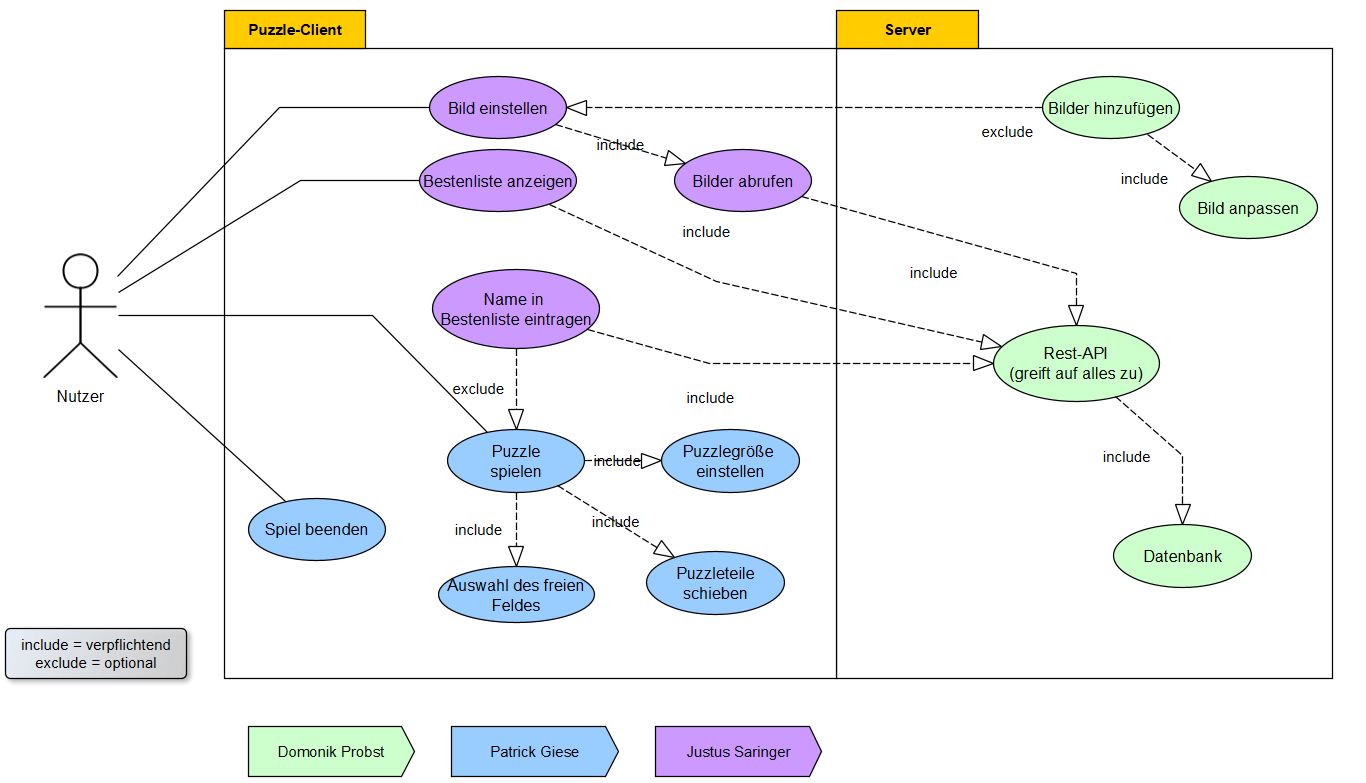
# Visionen und Ziele

|  |  |  |  |
| --- | --- | --- | --- |
| Feature-ID | Priorität | Beschreibung | Aufwand |
| 01 | Hoch | Das Auswählen eines freien Feldes zu Anfang des Spiels | Mittel |
| 02 | Hoch | Eine Bildauswahl aus der Bilderliste durchführen | Niedrig |
| 03 | Mittel bis niedrig | Hinzufügen von Bildern zur Bilderliste | Niedrig |
| 04 | Hoch | Puzzlegröße einstellen | Niedrig |
| 05 | Niedrig | Einstellgrenzen der Puzzlegrößen | Niedrig |
| 06 | Hoch | Puzzleteile jeweils einzeln zum freien Feld schieben | Hoch |
| 07 | Hoch | GUI mit jeweiligen Buttons und genannten Features | Hoch |
| 08 | Mittel | Das Anzeigen der Bestenliste (Global) | Hoch |
| 09 | Mittel | Die Möglichkeit, nach einem erfolgreichen Abschluss eines Spiels, einen Namen in die Bestenliste einzutragen | Mittel |
| 10 | Niedrig | Bestenliste ist Global | Hoch |
| 11 | Hoch | Das Spiel muss eine Verbindung zum Server aufbauen können, sodass das Spiel gestartet sowie die Spielergebnisse übermittelt werden können. | Hoch |

# Anforderungen an Ihr System

## Use-Cases

Das Use-Case-Diagramm des Puzzle-Programms besteht aus zwei gleichwertigen Systemen. Entsprechend dem Server und dem Client. Der Client beinhaltet das Spiel sowie die dazugehörige Spiellogik. Der Server stellt auf Abruf die Bestenliste und die spielbaren Bilder aus der Datenbank zur Verfügung.



## Risiken

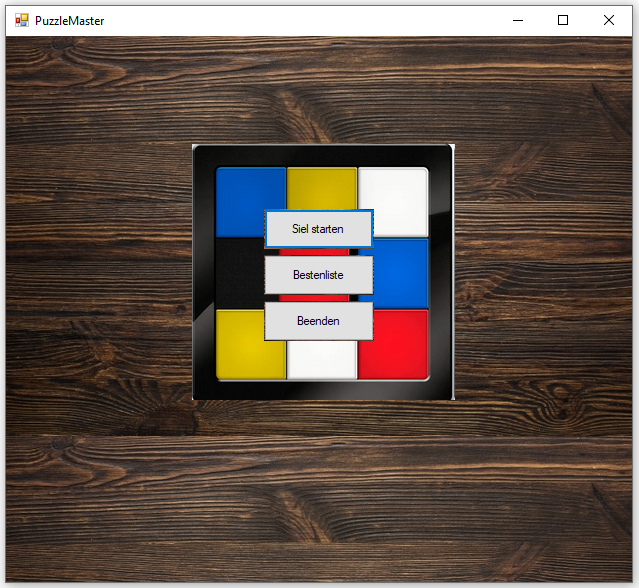
**Liste mit Risiken und Nebenwirkungen:**

Es werden Tools eingesetzt zu denen, bei den Projektmitgliedern, noch keine Erfahrungswerte vorliegen. Dadurch kann es passieren, dass im Nachhinein noch Änderungen an zentralen Komponenten durchgeführt werden müssen. Dies könnte ein erhöhtes Maß an Planungs- und Implementierungszeit erfordern, wodurch eine rechtzeitige Fertigstellung der Software gefährdet ist. Explizit handelt es sich dabei um folgende Komponenten, in dem die Teammitglieder bisher wenig bis keine Erfahrung gesammelt haben.

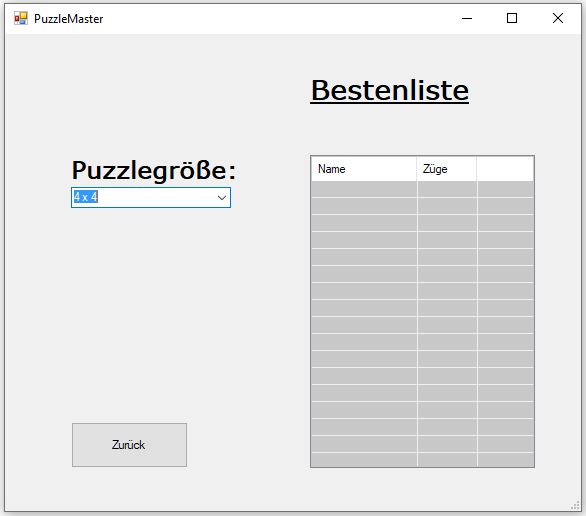
* ASP.NET zur Kommunikation über eine REST-Schnittstelle
* System.Data.SQLite.Core und Dapper für den Verbindungsaufbau und die Abfrage der Daten aus einem Datenbankmanagementsystem

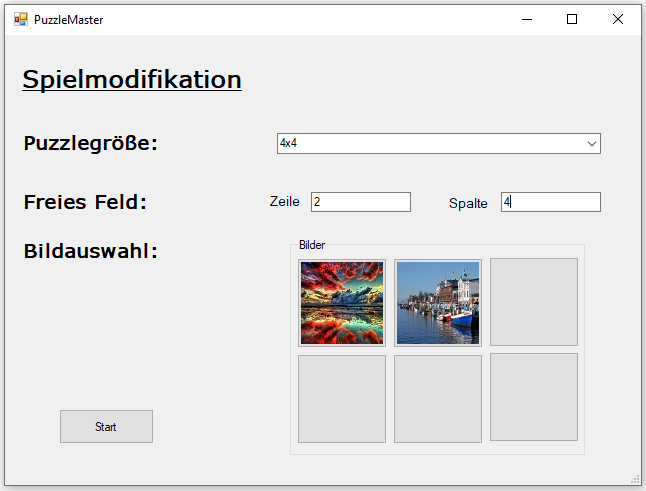
Nach einer Evaluierung der oben genannten Tools gehen wir davon aus, dass diese den Anforderungen entsprechend sind und somit das Gesamtrisiko als gering bewertet werden kann.

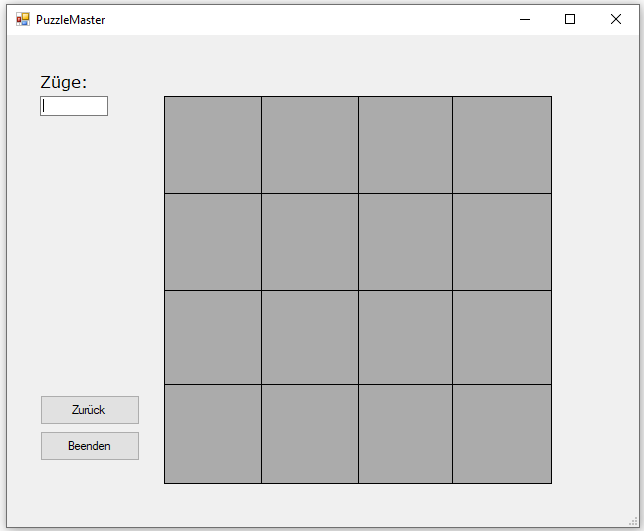
## GUI



Der Startbildschirm besteht aus drei Knöpfen. Ein Knopf mit der Aufschrift “Spiel starten”, welcher auf das Kontextmenü der Spielemodifikationen verweist. Der mit “Bestenliste” beschriftete Knopf öffnet die Bestenliste. Der letzte Knopf “Beenden” beendet, dem Kontext entsprechend, das gesamte Spiel.

Die Bestenliste besteht aus drei Modulen. Aus einem “Zurück” Knopf mit welchem man zurück auf den Startbildschirm gelangt. Deutlich über dem Zurück-Knopf ist die interaktive Puzzlegröße. Hier muss der Nutzer die gewünschte Puzzlegröße auswählen um die entsprechende Liste zu erhalten. Das letzte Modul ist die Liste selbst. Hier wird die Bestenliste passend zur ausgewählten Puzzlegröße angezeigt. Diese beinhaltet den Namen des Spielers und seine Züge. Die Reihenfolge geht von oben nach unten, damit steht oben der beste Spieler.

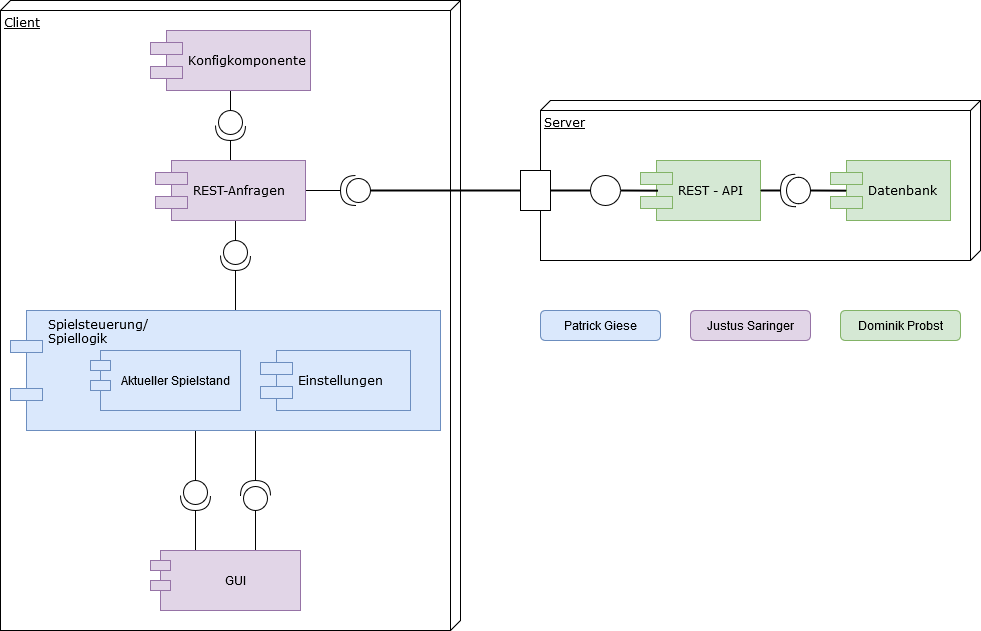
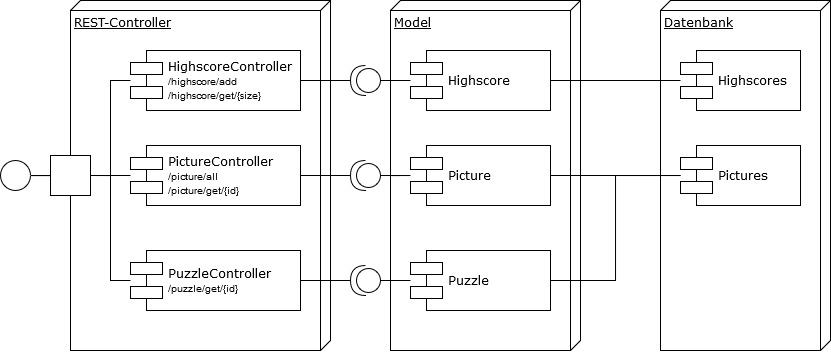
Wurde “Spiel Starten” im Startbildschirm gedrückt gelangt man zu den Spielemodifikationen. Hier werden sämtliche Einstellungen des Spiels durch den Spieler festgelegt. Unter anderem wird hier die Puzzlegröße eingestellt sowie einzeln die Position des freien Feldes über die Eingabe von Spalte und Zeile bestimmt. In der Bildauswahl kann das gewünschte Bild per Klick ausgewählt werden. Mit dem “Start” - Knopf wird das Spiel mit den eingestellten Spielemodifikationen gestartet und auf das Spielfenster verwiesen.



Oben links wird die Anzahl der Züge angezeigt, die der Spieler bisher getätigt hat. In der Mitte ist das Spielfeld, auf welchem sich das interaktive Puzzle befindet. Unten links sind zwei Knöpfe positioniert, welche mit “Zurück” und „Beenden“ beschriftet sind. Mit dem Zurück-Knopf gelangt man wieder in das Startmenü, wodurch man hingegen mit dem Beenden-Knopf das gesamte Spiel schließt.

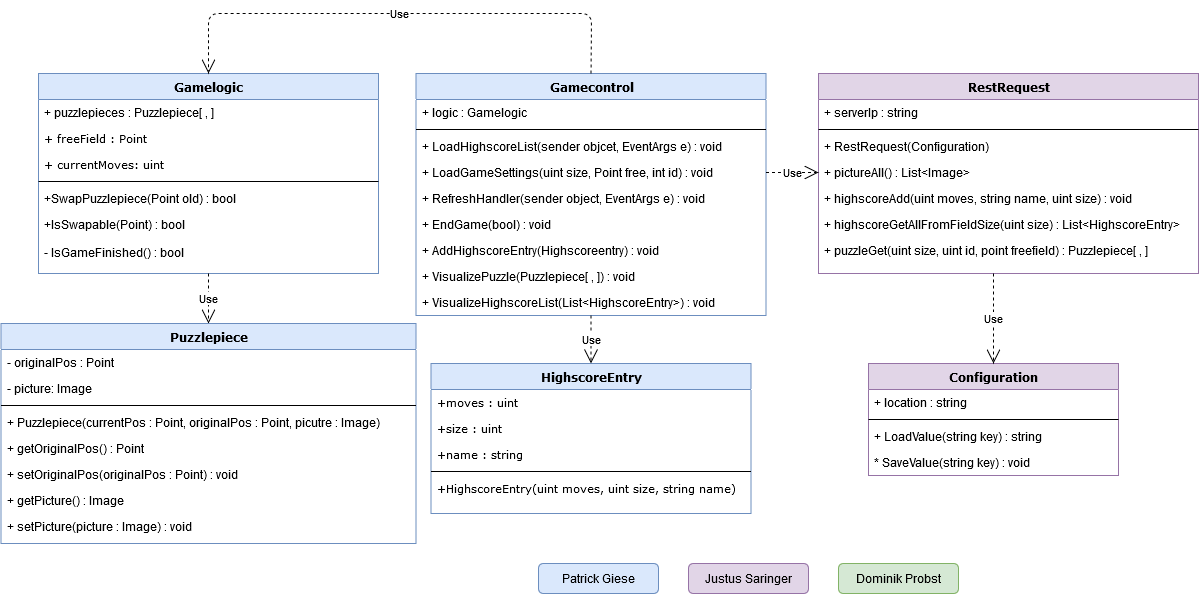
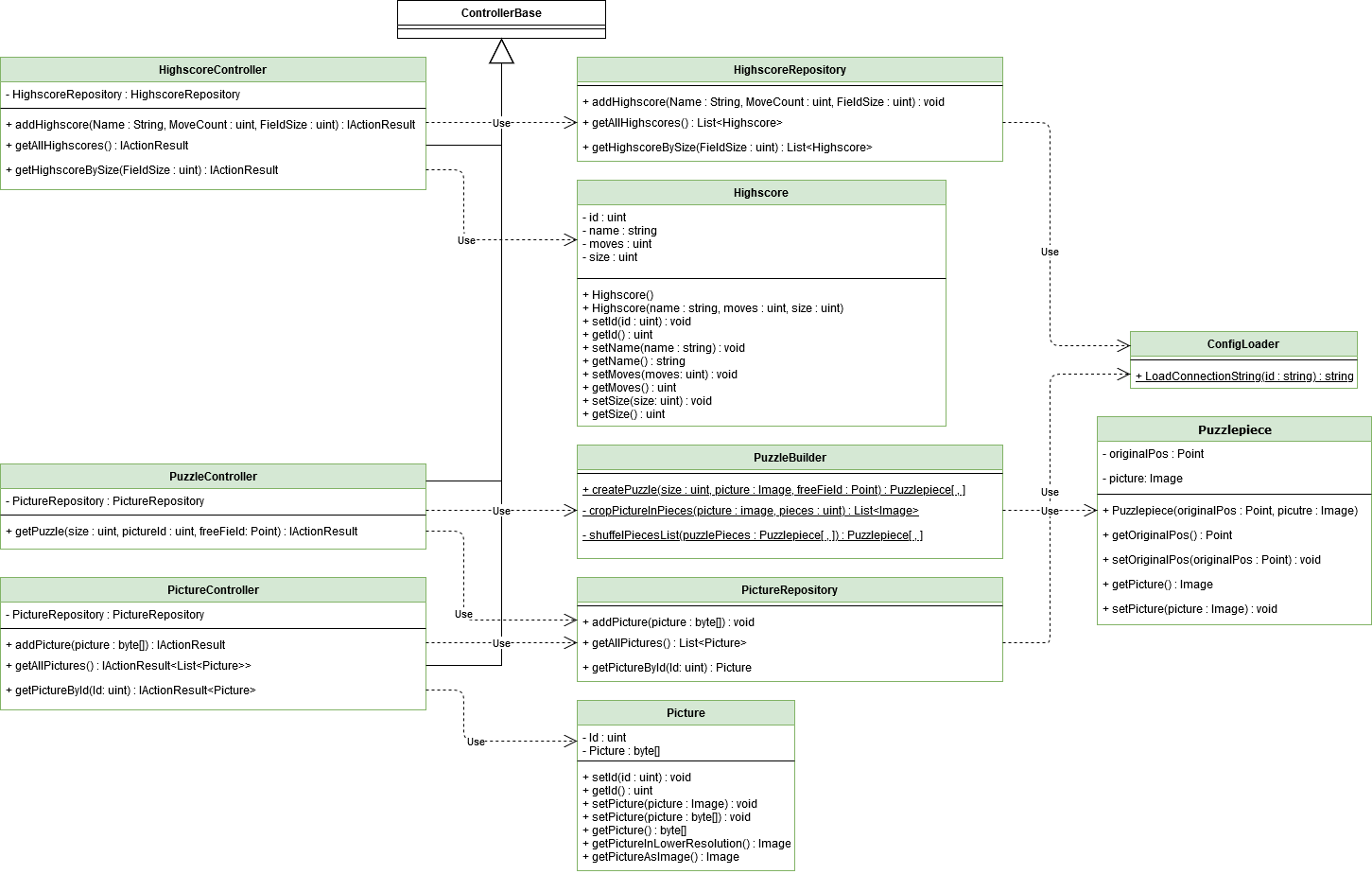
# Realisierung

## Allgemeines



## Interne Schnittstellen

|  |  |  |  |
| --- | --- | --- | --- |
| **Komponente** | **Funktion** | **Beschreibung** | **Werte / Rückgabewerte** |
| REST-Anfragen | RestRequest | Ruft die Server-IP über die Configuration auf | nichts / string |
| Rest-Anfragen | pictureAll | Sendet Anfrage an Server und gibt eine Liste an Bildern zurück | nichts / List<Image> |
| Rest-Anfragen | highscoreAdd | Fügt der Datenbankbestenliste einen neuen Eintrag hinzu | uint moves, string name, uint size / nichts |
| Rest-Anfragen | highscoreGetAllFromFieldSize | Fragt entsprechend zur eingegeben Größe des Spielfeldes die Bestenliste am Server nach und gibt diese zurück | uint size / List<HighscoreEntry> |
| Rest-Anfragen | puzzleGet | Fragt mit entsprechenden Anforderungen den Server nach einem Puzzle und gibt dieses zurück | uint size, uint id, point freefield / Puzzlepiece[ , ] |
|  |  |  |  |
| Spielsteuerung | LoadHighscoreList | verweist auf Methode (highscoreGetAllFromFieldSize) | uint size / nichts |
| Spielsteuerung | LoadGameSettings | Erstellt das Spiel mit angebenen Einstellungen | uint size, Point freefield, int id / nichts |
| Spielsteuerung | RefreshHandler | Führt die Visualizekomponenten aus |  |
| Spielsteuerung | EndGame | Beendet ein erfolgreiches Spiel | bool / nichts |
| Spielsteuerung | AddHighscoreEntry | Fügt der Datenbankbestenliste einen neuen Eintrag hinzu mit Aufrufen der RestRequest-Methode | HighscoreEntry / nichts |
| Spielsteuerung | VisualizePuzzle | Zeigt das aktuelle Puzzle in der GUI an | Puzzlepiece[ , ] / nichts |
| Spielsteuerung | VisualizeHighscoreList | Zeigt die entsprechende Bestenliste an | List<HighscoreEntry> / nichts |
|  |  |  |  |
| Spiellogik | SwapPuzzlepiece | Wechselt das freie Feld mit dem angeklickten Feld | Point old / bool |
| Spiellogik | IsSwapable | Prüft ob das angeklickte Feld mit dem freien Feld wechselbar ist | Point / bool |
| Spiellogik | IsGameFinished | Prüft ob das Puzzle gelöst ist | nichts / bool |
|  |  |  |  |
| Konfigkomponente | LoadValue | Liest einen Wert aus der Konfigdatei aus | String / string |
| Konfigkomponente | SaveValue | Speichert einen Wert in die Konfigdatei | String / nichts |
|  |  |  |  |
| GUI |  | Über die Ereignisse/Events werden die unterschiedlichen Methoden der Spielsteuerung |  |



## Visual-Studio-Projektsetup

Das Projektsetup besteht aus zwei Projekten. Dabei definiert ein Projekt die Anwendung für den Client und das andere Projekt den REST-Server. Das Projekt, inklusive Mockup und Dummy Klassen, ist im [Gitlab](https://gitlab.rz.htw-berlin.de/programmierprojekt/wisi2020/grp08/-/tree/master/programm) der HTW Berlin zu finden.

Für das separate Arbeiten an dem Projekt werden Branches für das jeweilige Teammitglied angelegt und genutzt. Vorerst liefern die Dummy-Klassen im Masterbranch Daten zum Testen zurück. Die bearbeiteten Klassen werden nach gemeinsamen Codereviews in den Masterbranch zusammengeführt.

## Externe Schnittstellen

**REST-Anfragen**  
Um die Kommunikation zwischen Client und Server zu ermöglichen, wird eine REST-API verwendet. Diese implementieren wir mithilfe der ASP.NET API, dabei stellt die REST-API folgende Schnittstellen zur verfügung.

|  |  |  |  |
| --- | --- | --- | --- |
| **Schnittstelle** | **HTTP GET** | **Rückgabe (JSON)** | **Beschreibung** |
| /picture/all |  | List<Image> | Gibt alle Bilder aus der DB in verringerter Auflösung zurück |
| /picture/get/{id} | ID des Bildes | Image | Gibt Bild in nativer Auflösung zurück |
| /highscore/add | Anzahl an Züge,  Name des Spielers,  Spielfeldgröße als einzelne Zahl (z.B. 4) | 200 OK  400 Bad Request | Fügt einen Highscore hinzu |
| /highscore/get/{size} | Spielfeldgröße als einzelne Zahl (z.B. 4) | List<Highscore> | Gibt alle Highscores der Angefragten Spielfeldgröße zurück |
| /puzzle/get | ID des Bildes,  Position des freien Feldes,  Spielfeldgröße als einzelne Zahl (z.B. 4) | Puzzlepiece[ , ] | Erstellt das Spielfeld anhand der übergebenen Werte und gibt das Puzzle zurück |

Auf eine REST-Anfrage wird eine Antwort, bestehend aus einem Statuscode und ggf. einem Objekt zurückgeliefert. Dabei beschränken wir uns auf die üblichen und häufig genutzten Statuscodes (200, 400, 404, 500).

**Datenbank**

Zur Speicherung der Daten wird eine SQLite Datenbank genutzt. Dazu wird die System.Data.SQLite.Core und Dapper benötigt. Als Schnittstelle dienen Repository Klassen, welche auf die Datenbank zugreifen.

# Test und Implementierungsphase

Das Projekt inklusive Test- bzw. Dummy Klassen ist im [Gitlab](https://gitlab.rz.htw-berlin.de/programmierprojekt/wisi2020/grp08/-/tree/master/programm) der HTW Berlin zu finden.

# Lizenz

Die Software wird unter der [MIT-Lizenz](https://de.wikipedia.org/wiki/MIT-Lizenz) veröffentlicht.