Software Praktikum (SoPra) – FS18

Milestone 1 – Assignment

**Group 15**

**Alther Robert**, robert.alther@uzh.ch (16-711-384)

**Bader Alena**, alenajacquelineemelie.bader@uzh.ch (15-934-615)

**Brun Yulia**, yulia.brun@uzh.ch (16-700-536)

**Raghupathy Abirami**, abirami.raghupathy@uzh.ch (16-730-368)

**Toedtli Laura**, laura.toedtli@uzh.ch (09-920-257)

1. Contents

[**1. Domain Model and User Stories 2**](#_30j0zll)

[*1.*](#_1fob9te) *Domain Model 2*

[*2.*](#_2et92p0) *User Stories 4*

[**2. User Interface Design – first Draft 13**](#_tyjcwt)

[**3. Risk Analysis 19**](#_2jxsxqh)

[**4. Project planning until Milestone 2 22**](#_z337ya)

Illustration Directory

[Figure 1: Domain Model. Top: Overview of all interactions. Bottom: Detailed view on interactions between child classes of Position 3](#_3znysh7)

[Figure 2: Welcome page 13](#_3dy6vkm)

[Figure 3: Creating player 13](#_1t3h5sf)

[Figure 4: Creating new game - Choose name 14](#_4d34og8)

[Figure 5: Creating a new game - Choose path to ElDorado and level of difficulty 14](#_2s8eyo1)

[Figure 6: Join an existing game 15](#_17dp8vu)

[Figure 7: Waiting room 15](#_3rdcrjn)

[Figure 8: Start of a new game 16](#_26in1rg)

[Figure 9: Use ActionCard 16](#_lnxbz9)

[Figure 10: Market board 17](#_35nkun2)

[Figure 11: Stock 17](#_1ksv4uv)

[Figure 12: End of the game 18](#_44sinio)

[Figure 13: Project in a Gantt-Chart 24](#_1y810tw)

# Domain Model and User Stories

## Domain Model

Based on the description of the game “ElDorado”, its physical appearance and discussions in the group, following entities/roles were identified:

**Game**: The abstract over all concept of the physical game (cards, playing pieces, board), its rules (interactions), actors (players) and gameplay (positions, history).

**Board**: Encloses part of the physical game (fields, blockades) and its rules (how to move a playing piece).

**Player**: Driving force of most actions during gameplay (card and playing piece movement).

**PlayingPiece:** Marker for the player on the board.

**Field:** Building blocks of the physical game and part of its rules (how to move onto field).

**Blockade:** Representation of the blockades and its rules.

**Card:** Representation of the physical cards and their capabilities. There is a distinction between ExpeditionCards, ActionCards and Jokers.

**ExpeditionCard:** Cards which have a symbol and a power.

**ActionCard:** Cards which trigger some additional action.

**Joker:** Card which is either a yellow, green or blue ExpeditionCard.

**Position**: Manifestation of the possible locations of the cards during the game. Includes: Selection, Hand, DrawPile, DiscardPile, Graveyard, Market, Stock.

**Selection**: Position a card is put onto when the player wants to use it.

**Hand**: Position of Cards which can only be seen by Player in charge of Hand. These cards can be put into Selection.

**DrawPile**: Position where Hand-Cards come from.

**DiscardPile**: Position where Cards go when they were used, discarded or bought.

**Graveyard**: Designated position for sacrificed cards.

**Market**: General term for piles on marketboard.

**Stock**: General term for piles next to marketboard which abide special buying conditions.

It was concluded that a move 1) always involves at least one card and 2) at least one card switches position during a move. Because of this understanding following artificial entities were imagined:

**History**: Collection of Move’s in chronological order which are relevant for all player (buying of cards, moving of playing piece, use of ActionCard).

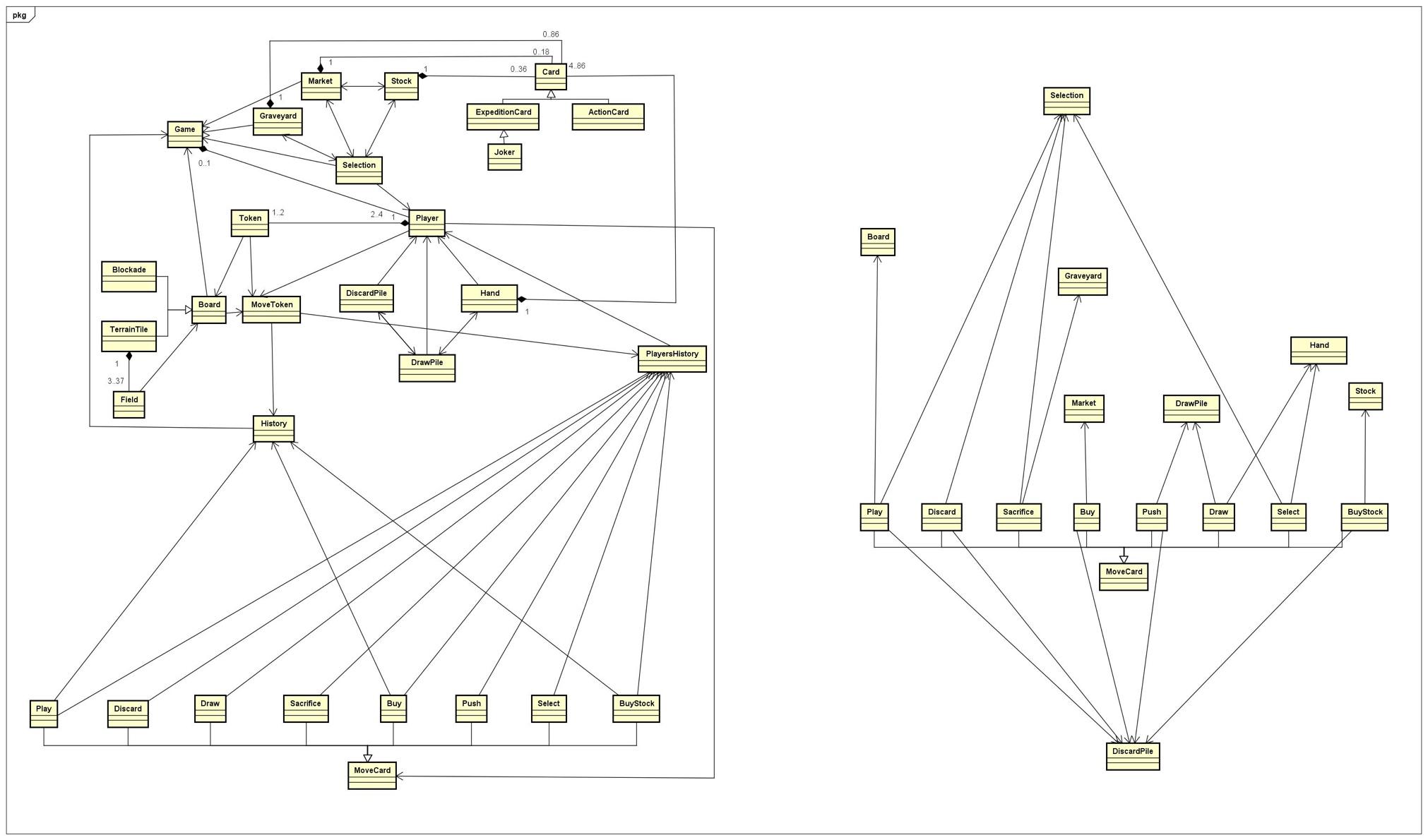
**PlayerHistroy**: Collection of Move’s in chronological order which only concern the player and can be undone.

**Move**: Manifestation of the location change of a playing piece or the position change of a card. Refers to both MoveCard and MoveToken.

**MoveCard**: Storage unit of a position change of a card.

**MoveToken**: Storage unit of a location change of a playing piece on board.

Depicted in Figure 1 the domain model of the described entities/roles and their interactions. The diagram at the bottom gives an overview while the one on the top of the page gives a detailed look upon the different position changes (MoveCard) that make sense.



*Figure 1: Domain Model. Top: Overview of all interactions. Bottom: Detailed view on interactions between child classes of Position*

## User Stories

Following are the user stories to our project description. They depict the requirements for our project. They are based on the El Dorado game and they represent the rules of the game among other things. The user stories ordered by role. Each user story has an assigned ID. The limitations to the user stories are listed separately and are also assigned an ID. These IDs are referenced in the Limitation section of a user story.

|  |  |  |
| --- | --- | --- |
| **ID** | **Description** | **Limitation** |
| L0XX | Existence |  |
| L001 | Game must exist |
| L002 | Board must exist |
| L003 | Player must exist |
| L004 | Positions must exist |
| L005 | Field must exist |
| L006 | Card must exist |
| L1XX | Concerning the game |  |
| L100 | Game must have at least two players |
| L101 | Game can’t have more than four players |
| L102 | Game must have a board |
| L103 | Game must have cards |
| L104 | Game must have positions |
| L2XX | Concerning the board |  |
| L201 | Board must have fields |
| L202 | Board must have starting fields |
| L203 | Board must have mountains and caves |
| L204 | Board must have final fields |
| L205 | Board must have base camp fields |
| L206 | Board must have expedition fields |
| L3XX | Concerning the field |  |
| L301 | Field must be adjacent to move onto it |
| L302 | The power value of the buying card must be equal or higher than the power of the field |
| L303 | The symbol on the buying card must match the fields symbol |
| L304 | The field can’t be occupied |
| L305 | Playing piece can’t move onto mountain nor cave fields |
| L306 | A playing piece must be associated with a field after initialization |
| L4XX | Concerning the player |  |
| L401 | It must be the player’s turn |
| L402 | Player can perform more than one action in a turn |
| L403 | Player must have a playing piece |
| L404 | Player must have a hand, draw pile and discard pile |
| L405 | Player can only have access to base cards at the beginning of turn |
| L406 | Player can’t undo draw actions |
| L407 | Player can only undo actions during his/her turn |
| L408 | Player is removed from the game if time is up |
| L409 | Player is removed from game if the player is disconnected |
| L5XX | Concerning the positions |  |
| L501 | Market board must have a vacant spot |
| L502 | Card of interest can only be bought if the value of the buying cards matches the price of the card of interest |
| L503 | Only one card can be bought per turn |
| L504 | A player’s empty draw pile is refilled by the cards from the same players discard pile |
| L505 | A position can contain no or several cards |
| L506 | There is only one selection, graveyard, market and stock per game |
| L6XX | Concerning the cards |  |
| L601 | A card must have a value |
| L602 | Cards can’t be removed from the game |
| L603 | Cards can’t be added to the game after initialization |
| L604 | The number of cards stays the same during the game |
| L605 | A card must be associated with a position after initialization |
| L606 | A card must be in the selection to be used |

Remark: Limitation ID L0XX refers to all limitations starting with L0, limitations with ID L1XX refers to all limitations starting with L1 and so on.

Following the user stories:

|  |
| --- |
| **ID101** |
| *User Story and Description* |
| As a Player I want to choose cards from my hand so that I can use these for an action.  The user must be able to move a card from his hand to the selection. |
| *Limitations* |
| L0XX, L102, L401, L404, L406, L407, L606 |
| *Acceptance Criteria* |
| Card objects can be selected by clicking on them once and the selected card object moves from Hand to Selection. |
| *Task* |
| Implement a method which will move a Card from Hand to Selection. |
| *Priority* |
| must |

|  |
| --- |
| **ID102** |
| *User Story and Description* |
| As a Player I want to move my playing piece by using an ExpeditionCard from the selection so that I can move closer to ElDorado.  The user must be able to move his playing piece on the board when the ExpeditionCard’s power is at least as high as the field’s power. |
| *Limitations* |
| L0XX, L102, L401, L404, L406, L407, L606 |
| *Acceptance Criteria* |
| An ExpeditionCard in the Selection can be used to move a playing piece by clicking on a valid field on the board. The ExpeditionCard shall stay in the Selection if the power of the card is higher than the power of the field. |
| *Task* |
| Implement the movement of a playing piece using ExpeditionCards. |
| *Priority* |
| must |

|  |
| --- |
| **ID103** |
| *User Story and Description* |
| As a Player I want to be able to play ActionCards so that I can use their action.  The user must be able to activate an ActionCard’s action. |
| *Limitations* |
| L0XX, L102, L401, L404, L406, L407, L606 |
| *Acceptance Criteria* |
| The user can trigger an ActionCards ability in the Selection by clicking the “Activate Card” button. |
| *Task* |
| Implement different actions for ActionCards. |
| *Priority* |
| should |

|  |
| --- |
| **ID104** |
| *User Story and Description* |
| As a Player I want to select the symbol of JokerCards so that I can use them to move my playing piece.  The user can choose the symbol of JokerCards in the Selection. |
| *Limitations* |
| L0XX, L102, L401, L404, L406, L407, L606 |
| *Acceptance Criteria* |
| A JokerCard in the Selection can be used as an ExpeditionCard of colour green, yellor or blue by clicking the “Activate Card” button and choosing the respective colour. As long as the JokerCard wasn’t used for moving a playing piece the colour can be changed. |
| *Task* |
| Implement a method for choosing the colour of JokerCard. |
| *Priority* |
| must |

|  |
| --- |
| **ID105** |
| *User Story and Description* |
| As a Player I want to move to adjacent fields so that I can reach El Dorado  The user is only allowed to move to fields which are adjacent to fields on which his/her playing piece is currently on. |
| *Limitations* |
| L0XX, L1XX, L201, L301, L304, L305, L306, L401, L403 |
| *Acceptance criteria* |
| The user clicks on the field to move his/her playing piece there. The system only allows this move if the field is adjacent to the current field and the Card in the Selection allows it. |
| *Task* |
| Implement a method that checks if a field is adjacent to the current field. |
| *Priority* |
| must |

|  |
| --- |
| **ID106** |
| *User Story and Description* |
| As a Player I want to sum the values of several cards up so that I can buy a new card.  It’s possible to combine several cards to perform a “buying action”. |
| *Limitations* |
| L0XX, L103, L401, L404, L502, L503, L601, L606 |
| *Acceptance criteria* |
| The user selects the hand cards he/she wants to use by clicking on them. The system displays the sum of the values on the players window. The cards move |
| *Task* |
| Implement a method of the Market that calculates the number of coins which have the Cards in the Selection. |
| *Priority* |
| must |

|  |
| --- |
| **ID107** |
| *User Story and Description* |
| As a Player I want to buy cards from the marketplace so that I can improve my deck.  The user can choose from different cards on the market board which then will be removed from the market and added to the discard pile of the user. |
| *Limitations* |
| L0XX, L103, L401, L404, L407, L502, L503, L506, L6XX |
| *Acceptance criteria* |
| The user clicks on the button “Marketplace” and then clicks on the desired card to buy that card from the marketplace. The system only allows the acquisition if the required value for the purchase is given and will and the purchased card to the players discard pile. |
| *Task* |
| Implement a method that moves a Card from Market to DiscardPile. |
| *Priority* |
| must |

|  |
| --- |
| **ID108** |
| *User Story and Description* |
| As a Player I want to see the played cards so that I can adapt my tactic.  Other players moves will be displayed after they end their turn. |
| *Limitations* |
| L0XX, L407 |
| *Acceptance criteria* |
| All the players can see a history of all played cards in their window during a game. |
| *Task* |
| Implement a method to saving and displaying the history of the game. |
| *Priority* |
| should |

|  |
| --- |
| **ID109** |
| *User Story and Description* |
| As a Player I want to switch between marketplace and playfield so that I can decide how to play.  The player can choose between moving his/her playing piece and buying a card. |
| *Limitations* |
| L0XX, L102, L104 |
| *Acceptance criteria* |
| The user clicks on the button “Marketplace” to switch to the marketplace or closes the tab to return back to the board. |
| *Task* |
| Implement buttons for Market and Board. |
| *Priority* |
| must |

|  |
| --- |
| **ID110** |
| *User Story and Description* |
| As a Player I want to see who’s currently playing so that I'll be ready to play.  It will be displayed whose turn it is |
| *Limitations* |
| L0XX |
| *Acceptance criteria* |
| The user whose turn it is can play. The other users cannot select any cards and cannot move their playing piece. The users can see a display that shows whose turn it is. |
| *Task* |
| Implement a method which should track turns and display the name of the player whose turn it is. |
| *Priority* |
| must |

|  |
| --- |
| **ID111** |
| *User Story and Description* |
| As a Player I want a clear signal when I need to play so that I won’t miss my turn.  The player will receive a signal when it’s his/her turn. |
| *Limitations* |
| L0XX |
| *Acceptance criteria* |
| A display shows a notification when it is the player’s turn. |
| *Task* |
| Implement a method which should display a notification when it is the player’s turn. |
| *Priority* |
| must |

|  |
| --- |
| **ID112** |
| *User Story and Description* |
| As a Player I want to end the turn so that the next player can play.  The player can decide when he/she wants to end his/her turn as long as the time is not up. |
| *Limitations* |
| L0XX, L401, L402 |
| *Acceptance criteria* |
| The user clicks on the button “ End of turn” to end his play. The system saves the users play and the game moves on to the next player. |
| *Task* |
| Implement a method that ends a player’s turn. |
| *Priority* |
| must |

|  |
| --- |
| **ID113** |
| *User Story and Description* |
| As a Player I want to have an overview of the playfield so that I can plan my moves.  The part of the board where the players playing piece is located will be displayed at all times. |
| *Limitations* |
| L0XX, L1XX, L2XX, L306 |
| *Acceptance criteria* |
| The playing board with all the playing pieces location will be visible for all player at any time. The user clicks on the button to zoom in or out to have an overview of the playing field.  The user can change the view on the board by zooming in and/or out (+, - button) or moving the viewable window by clicking in the respective direction on arrow-shaped buttons. |
| *Task* |
| Implement a method which will allow a player to zoom in and out and to move the viewable window of the Board. |
| *Priority* |
| should |

|  |
| --- |
| **ID114** |
| *User Story and Description* |
| As a Player I want to buy cards from the stock so that I can improve my deck.  The user will be allowed to buy cards from the stock if there is a free spot at the market place or if an action card allows it. |
| *Limitations* |
| L0XX, L103, L104, L401, L404, L407, L501, L502, L503, L506, L601, L606 |
| *Acceptance criteria* |
| The user clicks on a card from the stock to buy a card from there. The system only allows this acquisition and will add the purchased card to the players deck, if and only if a spot at the marketplace is free or the selected action card allows it. |
| *Task* |
| Implement a method for moving a Card from Stock to DiscardPile of the player. |
| *Priority* |
| must |

|  |
| --- |
| **ID115** |
| *User Story and Description* |
| As a Player I want to get my hand filled up to 4 cards at the end of my turn so that I can be ready for my next turn.  The hand of a player will be automatically filled up at the end of his/her turn. |
| *Limitations* |
| L0XX, L103, L04, L401, L404, L406, L504 |
| *Acceptance criteria* |
| The User clocks on the button” End of Turn”. The system clears the selection field and fills the hand cards of the user. If needed the cards will be shuffled and filled up to 4. |
| *Task* |
| Implement a method which will move Cards from the DrawPile of the player to his/her Hand so that there are four Cards in the Hand afterwards. |
| *Priority* |
| must |

|  |
| --- |
| **ID116** |
| *User Story and Description* |
| As a Player I want to keep cards so that I can use them later.  It is possible to keep cards in the hand after the end of turn. |
| *Limitations* |
| L0XX, L401, L404, L406 |
| *Acceptance criteria* |
| The user clicks the button “End of turn” while still having cards in his hand and not selected, to keep the cards and maybe play them in the next turn. |
| *Task* |
| Implement a method which will allow a player to end his turn and to keep the rest of the Cards in the Hand. |
| *Priority* |
| must |

|  |
| --- |
| **ID117** |
| *User Story and Description* |
| As a Player I want to remove blockades so that all Players can move forward.  Blockades will disappear permanently from the board when the player paid the removing price. |
| *Limitations* |
| L0XX, L102, L301, L302, L303, L401, L403, L404, L407, L601, L606 |
| *Acceptance criteria* |
| The user clicks on the blockade to free the path. The system allows this move only if the value of the blockade matches the value of the card and the blockade disappears permanently. The number of blockades removed by the user will be upped by one and the value of the blockade will be assigned to the user. |
| *Task* |
| Implement a method which will allow a player to remove blockades. |
| *Priority* |
| must |

|  |
| --- |
| **ID118** |
| *User Story and Description* |
| As a Player I want to choose from different terrain routes so that I can play in different environments  The user can choose from six paths to Eldorado (as described in the game description). |
| *Limitations* |
| L0XX |
| *Acceptance criteria* |
| The user clicks on the desired terrain route and the game will be played in that desired terrain with its level of difficulty. |
| *Task* |
| Implement a method which will allow a user to choose a path for the game. |
| *Priority* |
| must |

|  |
| --- |
| **ID119** |
| *User Story and Description* |
| As a Player I want to choose the number of players so that I can play with one to three friends.  The user can choose with how many players he/she wants to play. |
| *Limitations* |
| L0XX, L100, L101 |
| *Acceptance criteria* |
| In the creation of a new game, the user clicks on the button “Two Players” , “Three Players” or “Four Players” to choose the number of players in the game. |
| *Task* |
| Implement a method which will allow a user to choose with how many other users he/she wants to play. |
| *Priority* |
| must |

|  |
| --- |
| **ID120** |
| *User Story and Description* |
| As a Player I want to choose colour of my token so that I can play with my favourite colour.  The user can choose from four different colors for his playing piece. |
| *Limitations* |
| L0XX |
| *Acceptance criteria* |
| When joining an existing game, the user clicks on the desired colour for his/her playing piece and it will be assigned to his/her playername. The last user to join a game can’t choose a colour. |
| *Task* |
| Implement a method which will allows a user to choose the colour of his/her playing piece. |
| *Priority* |
| must |

|  |
| --- |
| **ID121** |
| *User Story and Description* |
| As a Player I want to choose what to do in my turn so that I can use my own strategy.  During his/her turn, the player can either buy a card, move his/her playing piece and/or play an action card. |
| *Limitations* |
| L0XX, L1XX, L201, L3XX, L401, L403, L404, L406, L407, L502, L503, L601, L606 |
| *Acceptance criteria* |
| The user chooses what he wants to do and clicks on the card and the desired button or field. |
| *Task* |
| Implement buttons for buy a card, make a move or play Action/JokerCard |
| *Priority* |
| must |

|  |
| --- |
| **ID122** |
| *User Story and Description* |
| As a Player I want to undo actions so that I can correct mistakes.  The player can undo actions as long as they do not involve drawing of cards. |
| *Limitations* |
| L0XX, L103, L104, L401, L402, L404, L406 |
| *Acceptance criteria* |
| The user clicks on the button “undo” to undo his last action. The “undo” button can be used several times. The undo-function applies to only actions during the current turn of a player and does not apply to drawing actions (and all actions before a drawing action). |
| *Task* |
| Implement the undo-function and a players history |
| *Priority* |
| could |

|  |
| --- |
| **ID123** |
| *User Story and Description* |
| As a Player I want to only be able to perform legal actions so that I am not disqualified.  The game will not allow illegal moves according to game description. |
| *Limitations* |
| L0XX |
| *Acceptance criteria* |
| The system does not allows illegal moves.  The user can’t buy more than one card per turn. The user can’t play cards which are not in the Selection. The user can’t reuse Cards from the Graveyard. The user can only move to adjacent fields using a single Card from the Selection. The user can only buy Cards from the Stock if there is a free spot on the market board or the “Transmitter” ActionCard is activated. |
| *Task* |
| Implement Methods to make sure that no illegal actions are possible. |
| *Priority* |
| must |

|  |
| --- |
| **ID124** |
| *User Story and Description* |
| As a Player I want to be able to look at the rules so that I can check my possibilities.  It will be possible to open a window with the game instructions at any time. |
| *Limitations* |
| L001 |
| *Acceptance criteria* |
| The user can click on the button “Game rules” to see the rules any time during the game and the rules will appear. |
| *Task* |
| Implement a button for displaying the rules. |
| *Priority* |
| could |

|  |
| --- |
| **ID125** |
| *User Story and Description* |
| As a Player I want to move onto a base camp field so that I could completely remove the cards which I don't want to use later.  If a player moves his/her playing piece onto a base camp field one card will be removed from the game completely. |
| *Limitations* |
| L0XX, L102, L103, L104, L201, L205, L301, L304, L306, L401, L403, L407, L506, L606 |
| *Acceptance criteria* |
| The user clicks on the hand card he/she wants to be removed from his/her deck, if he/she moves onto a base camp field and the card will be removed from the game completely. |
| *Task* |
| Implement a method that will move Cards from Selection to Graveyard. |
| *Priority* |
| must |

|  |
| --- |
| **ID126** |
| *User Story and Description* |
| As a Player I want to use a high-power card to move through several fields at once so that I could reach El Dorado faster.  It will be possible to move a playing piece through several fields (with the same color) if the sum of all powers of the fields is equal or lower than the played card. |
| *Limitations* |
| L0XX, L102, L103, L201, L3XX, L401, L402, L403, L404, L606 |
| *Acceptance criteria* |
| The user clicks one after another field to move several fields in the same play with one card. The system allows the move only if all the value of the fields is equal or lower than the played card and if the card and fields color matches.  When moving the playing piece, the used Card will stay in the Selection if its power is higher than the field’s. The Card can be reused with the reduced remaining power. |
| *Task* |
| Implement a method which will subtract the power of the Field from the power of the Card. |
| *Priority* |
| must |

|  |
| --- |
| **ID127** |
| *User Story and Description* |
| As a Player I want to be able to choose my name so that I can identify myself.  At the beginning of the game each player can enter his/her username. |
| *Limitations* |
| L001, L003 |
| *Acceptance criteria* |
| The User enters his/her Playername in the given Box and it will be saved. |
| *Task* |
| Implement a box where a user can enter his/her name and a method which will save it. |
| *Priority* |
| must |

|  |
| --- |
| **ID128** |
| *User Story and Description* |
| As a Player I want know the number of blockades and their value when I reach El Dorado so that I know if I win if two players reach El Dorado in the same round.  The blockades will be automatically evaluated if two or more players reach El Dorado in the same turn to identify the winner. |
| *Limitations* |
| L0XX, L102, L201 |
| *Acceptance criteria* |
| The User with the highest number of blockades or value will get a message that he won the Game, if two or more player reaches El Dorado in the same turn.  The user who first reached ElDorado wins. If two or more player reach ElDorado in the same turn then the number of blockades are compared. The user with the most blockades win. If two or more player reached ElDorado in the same turn and have the same amount of blockades then the player with the highest power blockade wins. |
| *Task* |
| Implement a method which calculates the value of blockades removed by a player. |
| *Priority* |
| must |

|  |
| --- |
| **ID129** |
| *User Story and Description* |
| As a Player I want the game to continue for the current turn when I or another player reaches El Dorado so that another player could reach El Dorado  The game is terminated when at least one player and all other players had their turns. |
| *Limitations* |
| L0XX, L102, L201, L204 |
| *Acceptance criteria* |
| The game ends when the turn in which the first player reached El Dorado is finished. |
| *Task* |
| Implement a method which will stop the game after all players finished their turns. |
| *Priority* |
| must |

|  |
| --- |
| **ID130** |
| *User Story and Description* |
| As a Player I want to create a new game, so that I could play with my friends.  A Player has an opportunity to create a new game. |
| *Limitations* |
|  |
| *Acceptance criteria* |
| A Player clicks a button “Create game”, and a new game is created. The player is first in the waiting list and starts then the play as a first player. |
| *Task* |
| Implement a button for creating a game. |
| *Priority* |
| must |

|  |
| --- |
| **ID131** |
| *User Story and Description* |
| As a Player I want to join an existing game, so that I don’t have to create a new game.  A Player has an opportunity to join an existing game. |
| *Limitations* |
| L0XX |
| *Acceptance criteria* |
| A Player clicks a button “Join game”. The player is added to the waiting list if not all players have already joined the game. If all players have joined, the game starts. The order of turns corresponds to the order of joining the waiting list. |
| *Task* |
| Implement a button for joining an existing game. |
| *Priority* |
| must |

|  |
| --- |
| **ID132** |
| *User Story and Description* |
| As a Player I want to reach El Dorado first, so that I could win the game.  A Player should move his/her playing piece onto one of El Dorado’s end-fields first in order to win the game (when no other players reach El Dorado in their turns). |
| *Limitations* |
| L0XX, L1XX, L204, L301, L302, L303, L304, L401, L403, L601, L606 |
| *Acceptance criteria* |
| A Player moves his/her playing piece onto one of three end-fields. Other players may finish their turns. The game is over. |
| *Task* |
| Implement end-fields of El-Dorado and a method which decides who is the winner of the game. |
| *Priority* |
| must |

|  |
| --- |
| **ID133** |
| *User Story and Description* |
| As a Player I want to use one time cards so that I could benefit from their actions.  A Player can use a one time card only once. Then it is completely removed from the game. |
| *Limitations* |
| L0XX, L1XX, L401, L403, L606 |
| *Acceptance criteria* |
| A Player plays a one time card. The card is removed from the game. |
| *Task* |
| Implement one time cards: they move from Selection to Graveyard. |
| *Priority* |
| must |

|  |
| --- |
| **ID201** |
| *User Story and Description* |
| As a TwoPlayers I want to move both my playing pieces so that I can reach El Dorado.  In a two-player game each player gets two playing pieces and can move both during one turn. |
| *Limitations* |
| L0XX, L100, L102, L103, L201, L3XX, L401, L402, L403, L404, L407, L601, L606 |
| *Acceptance criteria* |
| The player clicks on one of the two playing pieces with which he wants to make the next move. |
| *Task* |
| Implement a method which allows a player to choose which playing piece to move. |
| *Priority* |
| must |

|  |
| --- |
| **ID301** |
| *User Story and Description* |
| As a Game Designer I want 1366px \* 768px so that the most common desktop resolution is covered.  The window size will have the dimensions 1366px to 768px. |
| *Limitations* |
| L001 |
| *Acceptance criteria* |
| The system has a window in the dimension 1366px to 768px. |
| *Task* |
| Implement the window dimension 1366px to 768px. |
| *Priority* |
| must |

|  |
| --- |
| **ID302** |
| *User Story and Description* |
| As a Game Designer I want that the player cannot move on mountain fields so that the rules are not violated.  Mountain fields are not accessible by the players. |
| *Limitations* |
| L0XX, L102, L201, L203, L403 |
| *Acceptance criteria* |
| The system does not allow the player the move on a mountain field. |
| *Task* |
| Implement mountain fields as always occupied. |
| *Priority* |
| must |

|  |
| --- |
| **ID303** |
| *User Story and Description* |
| As a Game Designer I want that the player cannot move on fields with players on so that the rules are not violated.  A field can only be occupied by the most one playing piece at the same time. |
| *Limitations* |
| L0XX, L102, L201, L403 |
| *Acceptance criteria* |
| The system only allows the user to move onto a field, if and only if the field is not occupied by another playing piece. |
| *Task* |
| Implement a method which will check if a field is occupied if a player wants to move his/her playing piece onto it. |
| *Priority* |
| must |

|  |
| --- |
| **ID304** |
| *User Story and Description* |
| As a Game Designer I want the field to behave as indicated by the colour and number on it so that the action and feedback are consistent  A player can move his/her playing piece onto a field only if his/her played card has the same color and at least the same power or a played action card allows it. |
| *Limitations* |
| L0XX, L102, L201, L301, L304, L305, L306, L401, L403, L601, L606 |
| *Acceptance criteria* |
| The system only allows the user to move onto a field if the value and colour of the cards match or outrank the one of the field. |
| *Task* |
| Implement a method which will check if a Card in Selection has enough power and if it matches the colour of the field which a player wants to move onto. |
| *Priority* |
| must |

|  |
| --- |
| **ID401** |
| *User Story and Description* |
| As a Development Team I want to introduce time restrictions for a turn so that the game doesn't take forever  There is a time limit for every turn and the turn ends when the time ran out. |
| *Limitations* |
| L0XX, L408, L409 |
| Acceptance criteria |
| The system ends the turn of a player automatically when the time is up and moves on to the next player. |
| *Task* |
| Implement a method which will end the turn of a player automatically when the time is up. |
| *Priority* |
| could |

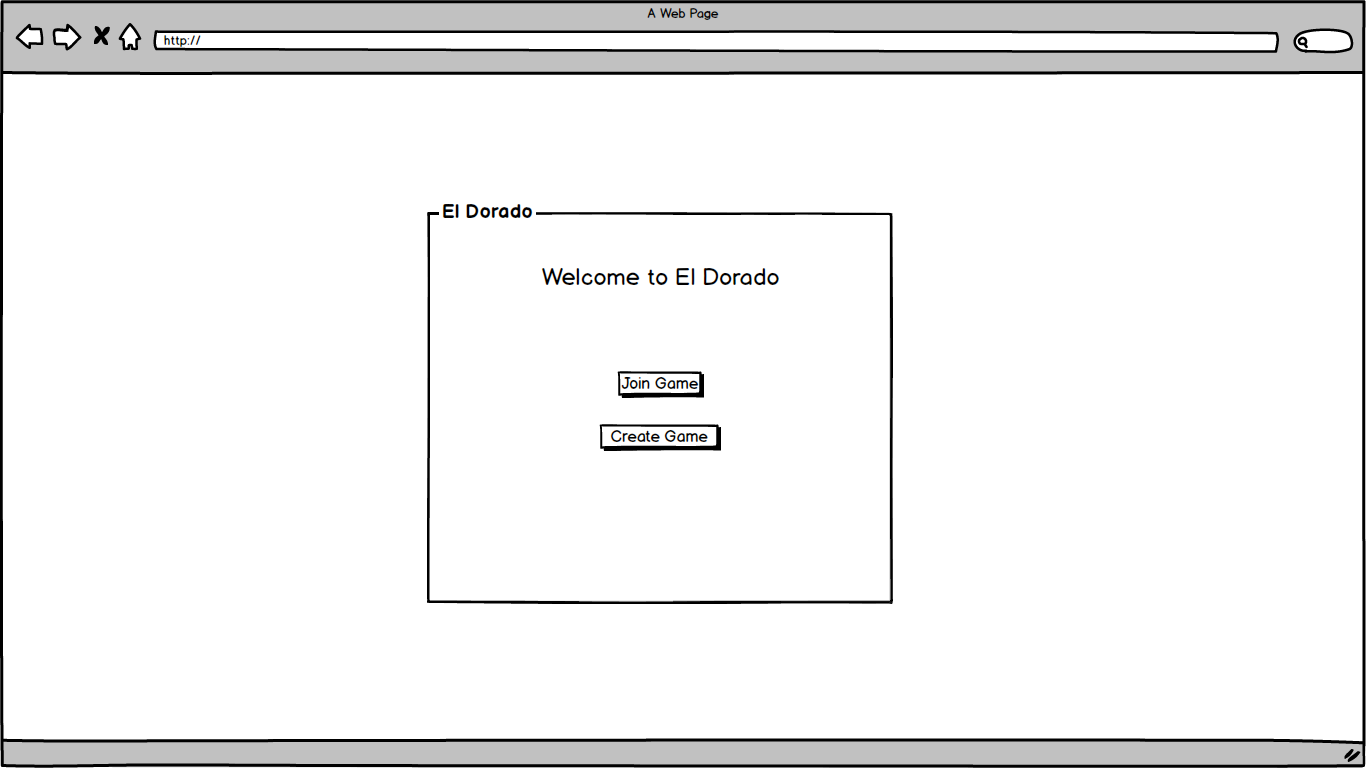
|  |
| --- |
| **ID402** |
| *User Story and Description* |
| As a Development Team I want the player only to move on existing fields so that the game doesn't take forever.  Non-existing fields can’t be accessed. |
| *Limitations* |
| L0XX, L102, L201, L301 |
| *Acceptance criteria* |
| The user clicks on any field to move. The system allows the move only if the neighbouring field exists. |
| *Task* |
| Implement the accessible fields. |
| *Priority* |
| must |

|  |
| --- |
| **ID403** |
| *User Story and Description* |
| As a Development Team I want to restrict the character input of the name of the player so that they don't put in a book  The input for the username is restricted by lengths of characters |
| *Limitations* |
| L001, L003 |
| *Acceptance criteria* |
| The user can type in a maximum of 15 characters as his name. |
| *Task* |
| Implement an input restriction for playrnames. |
| *Priority* |
| could |

# User Interface Design – first Draft

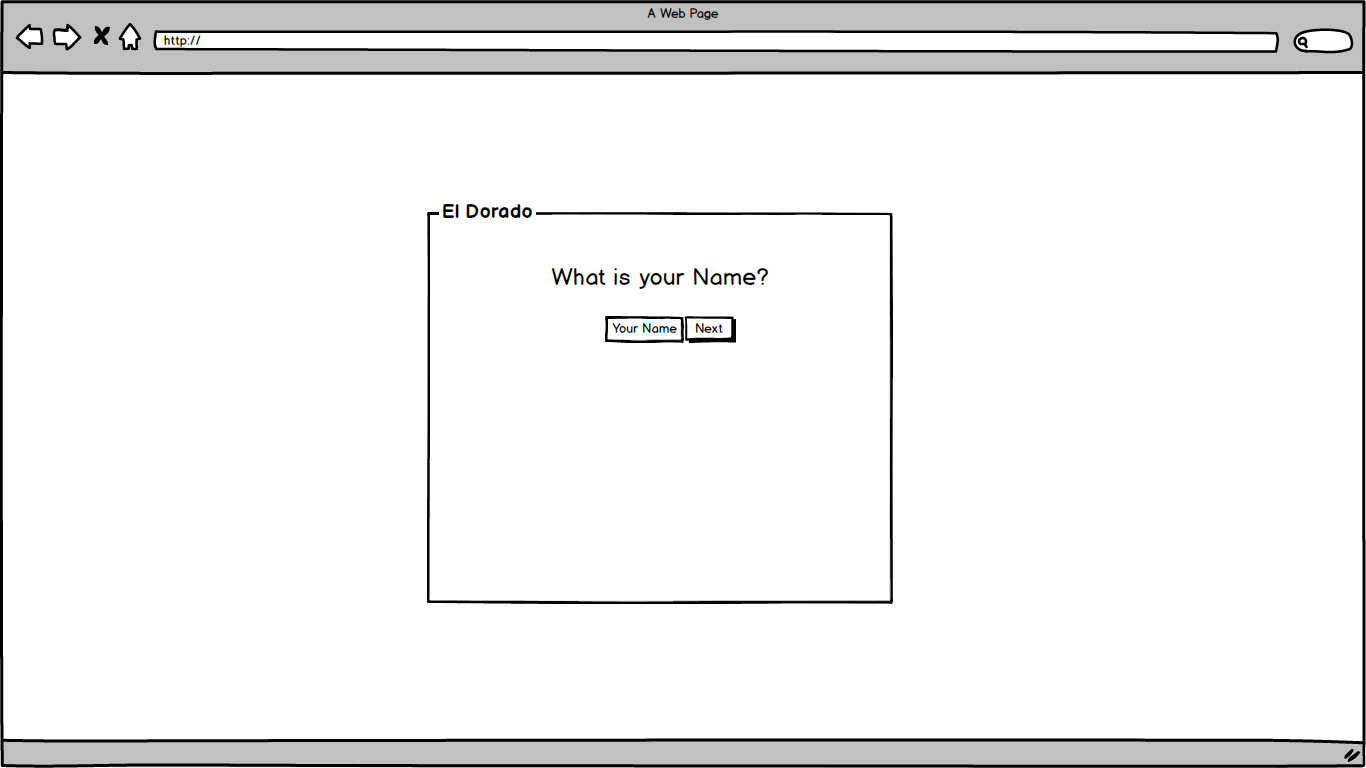
This chapter is concerned with the first draft of the physical appearance of the user interface for the web application “ElDorado”. Following the description and mock-ups for the various stages of a game play.

At the beginning the user can decide whether to join or create a game on the welcome page (Figure 2).



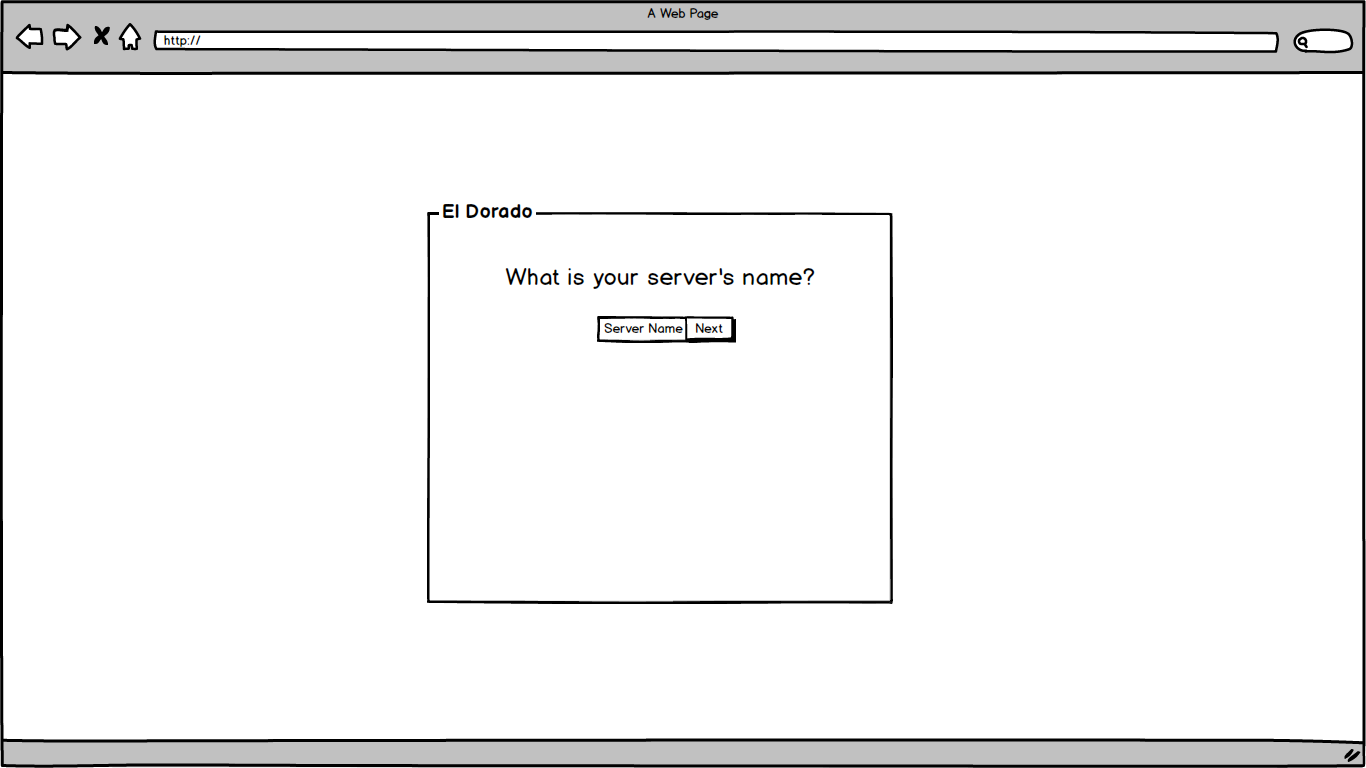
*Figure 2: Welcome page*

Independent of the choice on the welcome page, every user will choose a username as shown in Figure 3.

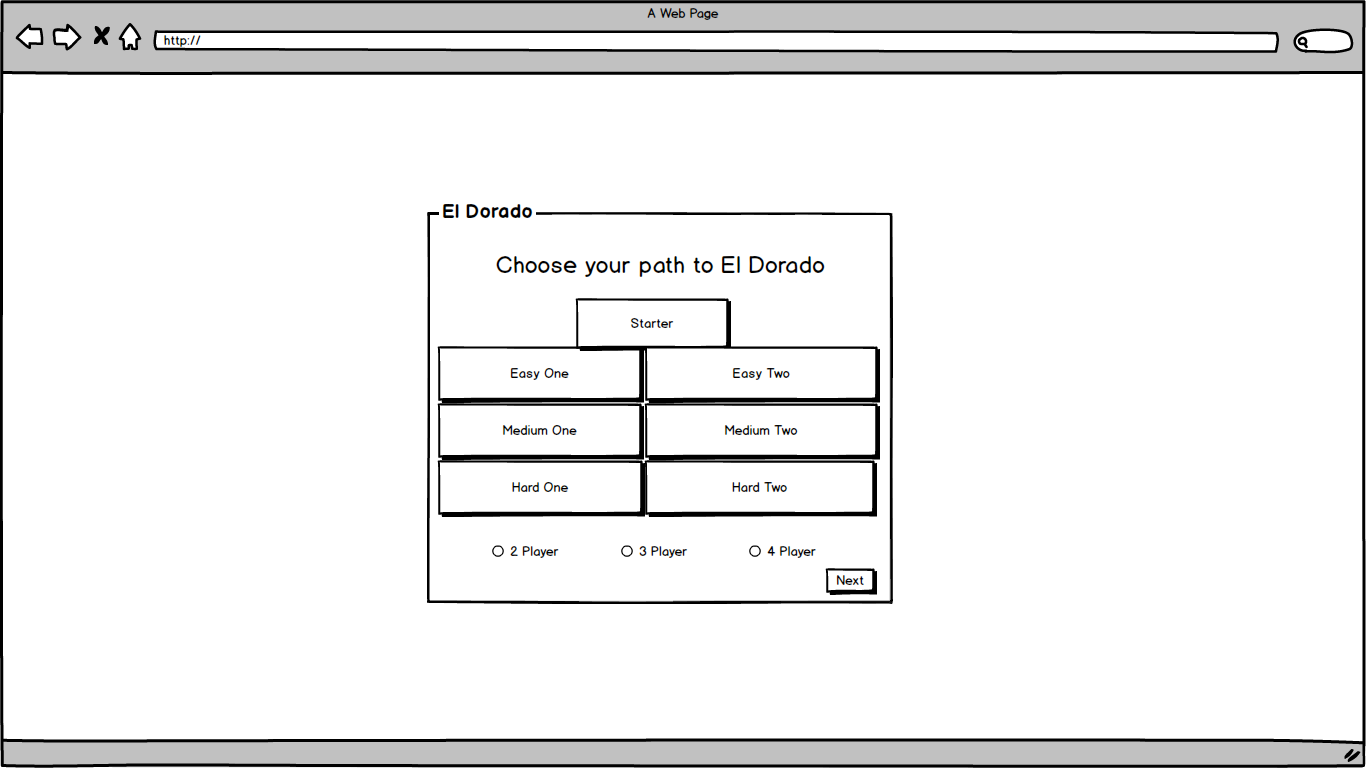


*Figure 3: Creating player*

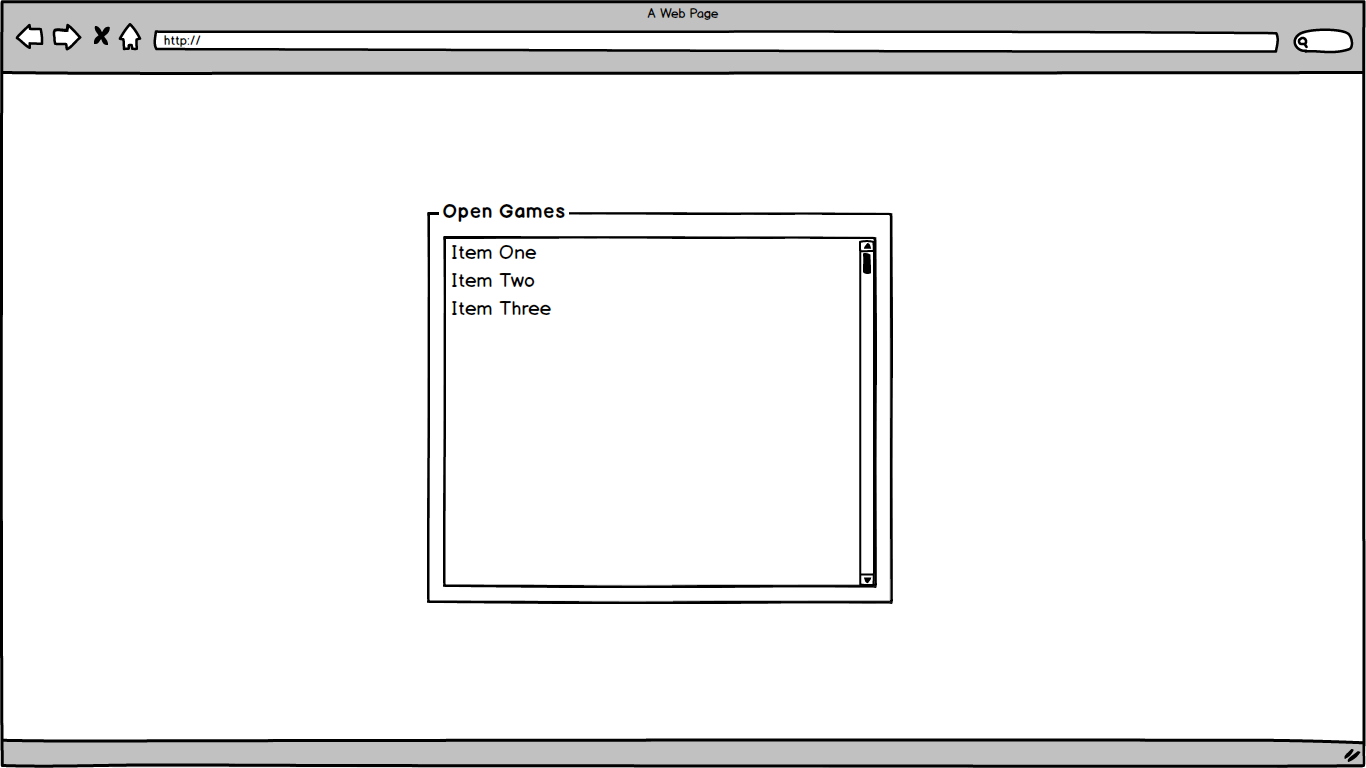
If a new game is created the user must choose a name for the server (Figure 4), so that other users can identify the server (Figure 6), choose a board and amount of players for the game (Figure 5). Only single selection is valid.



*Figure 4: Creating new game - Choose name*

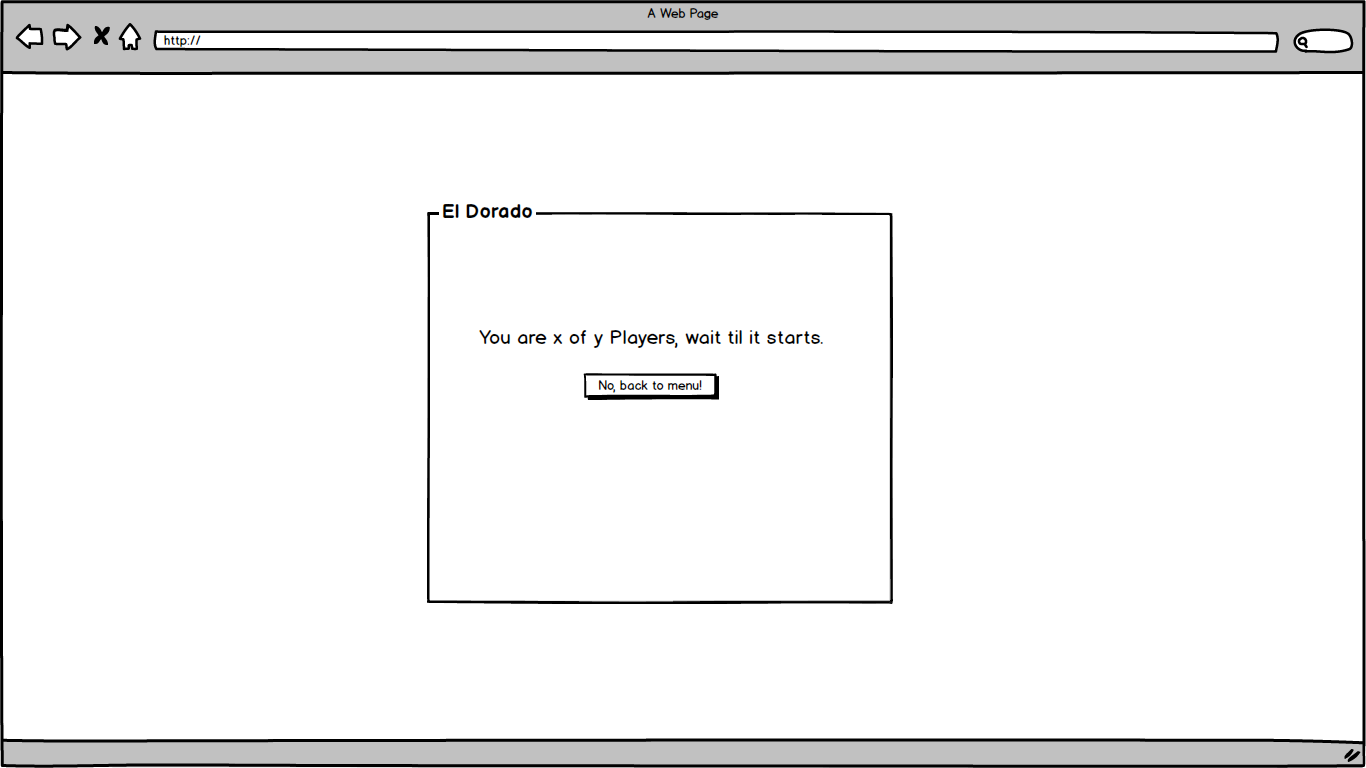


*Figure 5: Creating a new game - Choose path to ElDorado and level of difficulty*



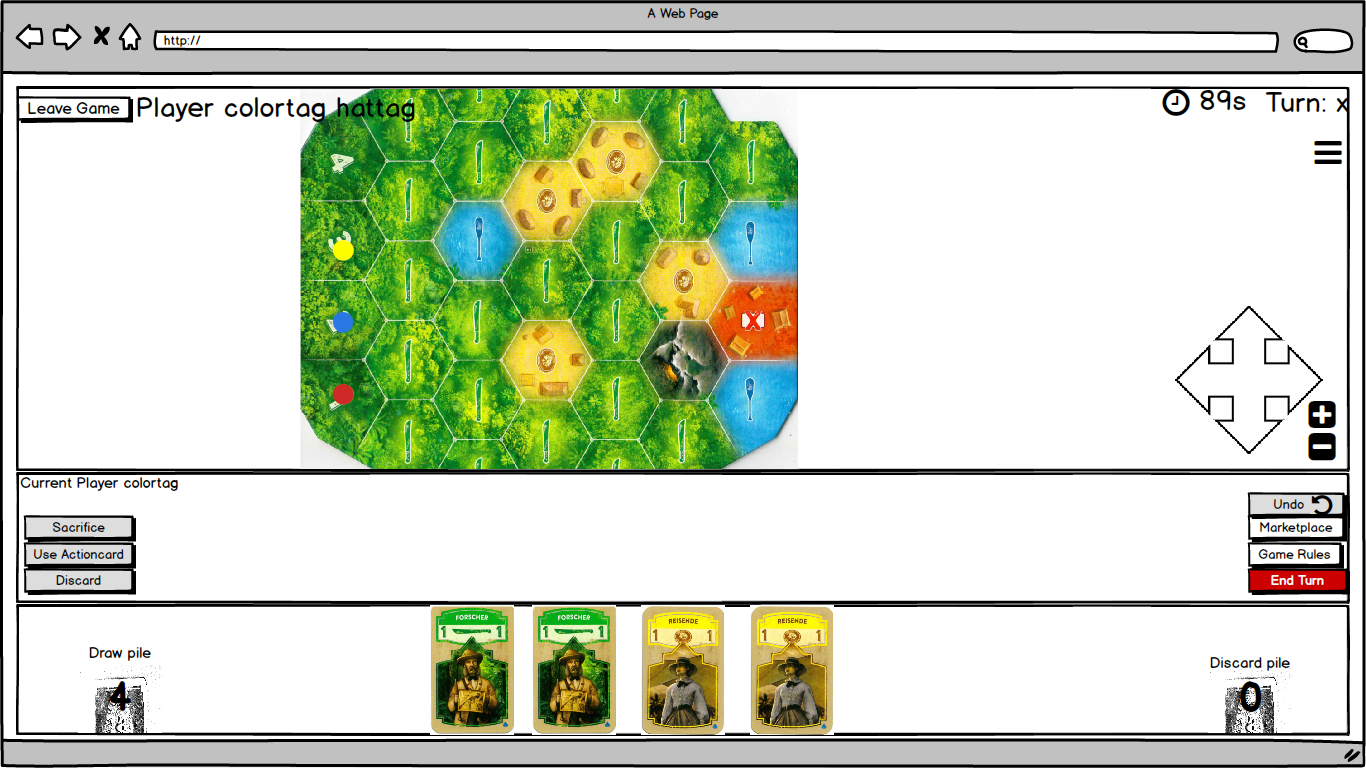
*Figure 6: Join an existing game*

A game starts only if the chosen number of players have joined (Figure 7). The waiting players are informed about how many they are and but still have the possibility to go back to the menu.



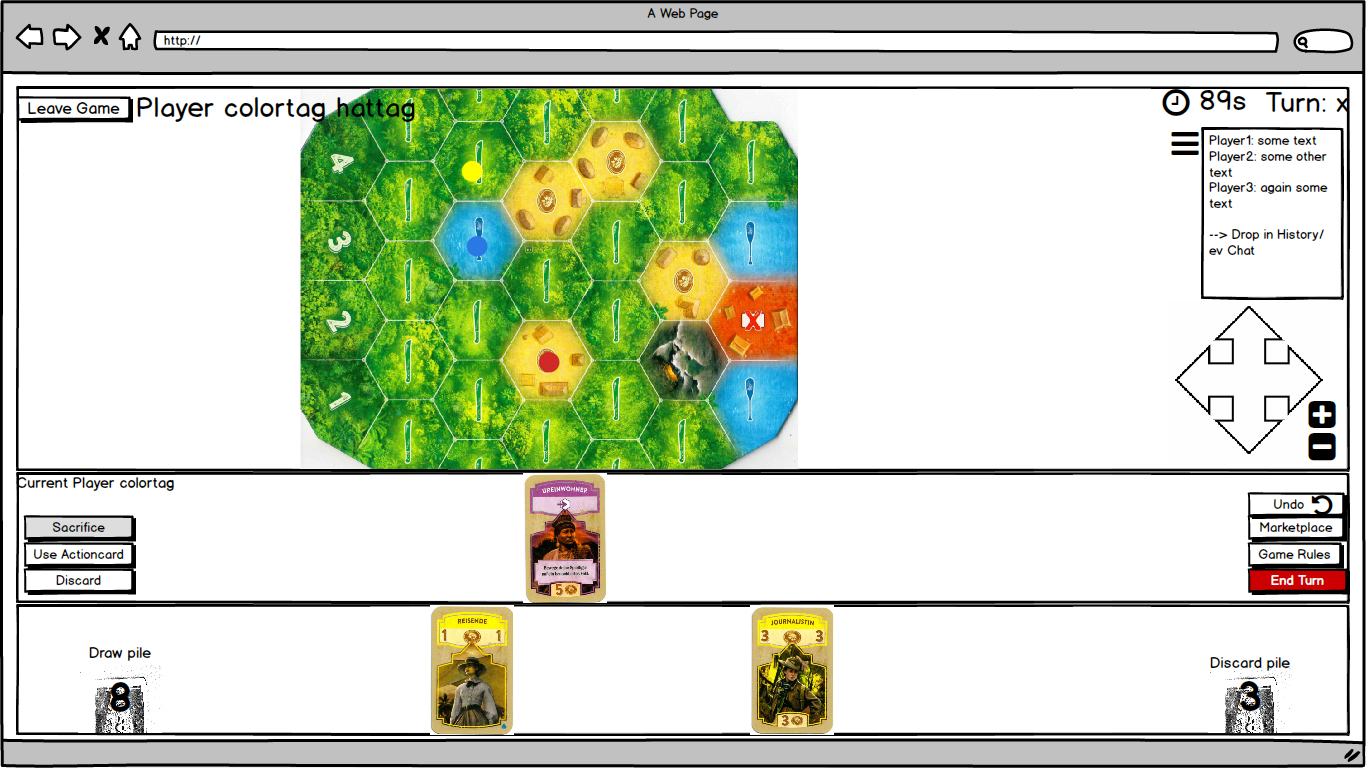
*Figure 7: Waiting room*

The initial state of the game will show the player his/her hand, discard pile, draw pile and the board. On the top left there is the “leave” button, the players name, and his/her color tag, if he/she’s the first player, he/she’ll get the hat tag. On the top right he/she sees the number of turns played and his/her remaining time to finish his/her turn. The view of the path can be manipulated by the controls on the right side. All this is shown in Figure 8.



*Figure 8: Start of a new game*

An ActionCard can be activated by clicking the “Use ActionCard” button on the left side. If he/she doesn’t want to activate it, he/she could discard it or use it as ½ coin to buy another card. To take the card back, the “Undo” button needs to be clicked. The user can look up the Game Rules by clicking the “Game Rules”. On the top right the user has the possibility to look up the game history of all players. To end his/her turn, he can click “End Turn”. All this is depicted in Figure 9.



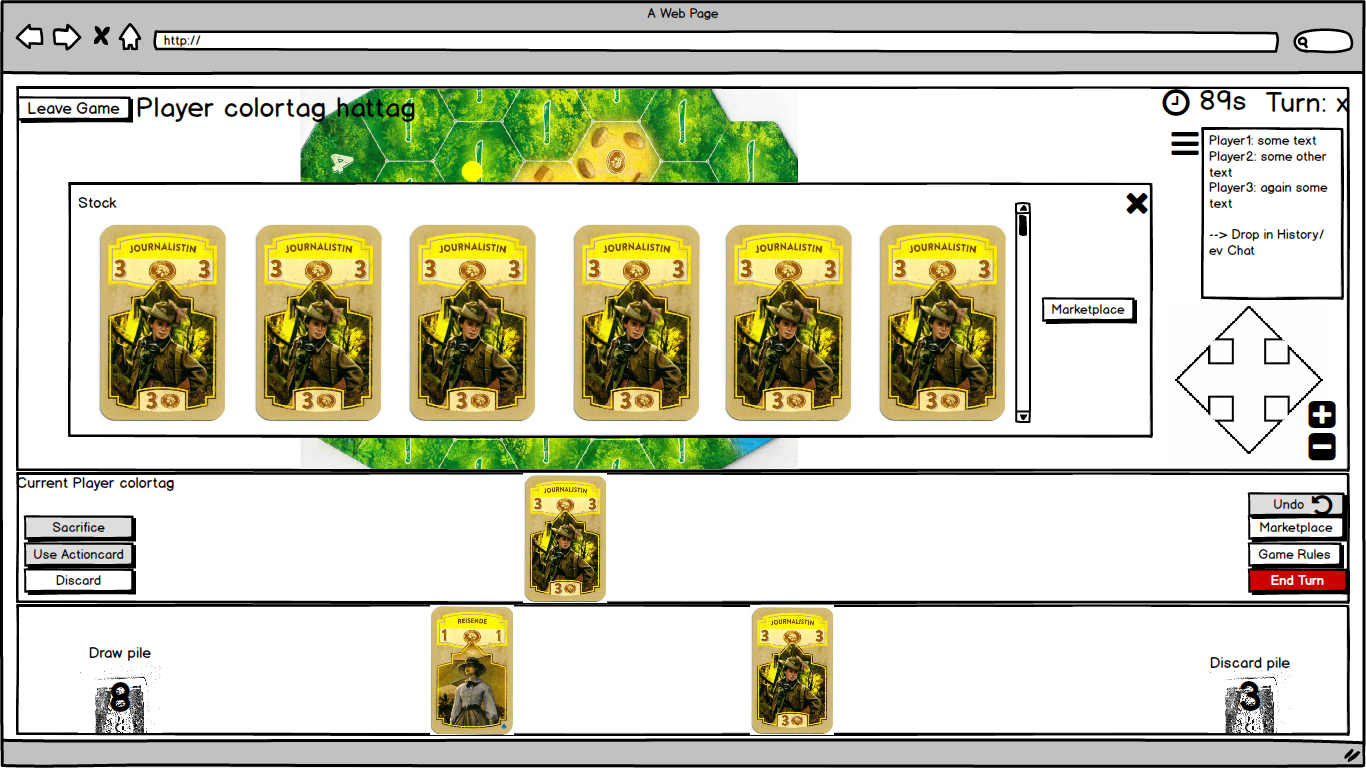
*Figure 9: Use ActionCard*

The marketplace button gives the user a pop-up with the Market and the option to go further to the Stock. In the Stock, there is the possibility to look at all 12 card piles via a scrollbar.

How these marketplace and stock will look are shown in Figure 10 and Figure 11.

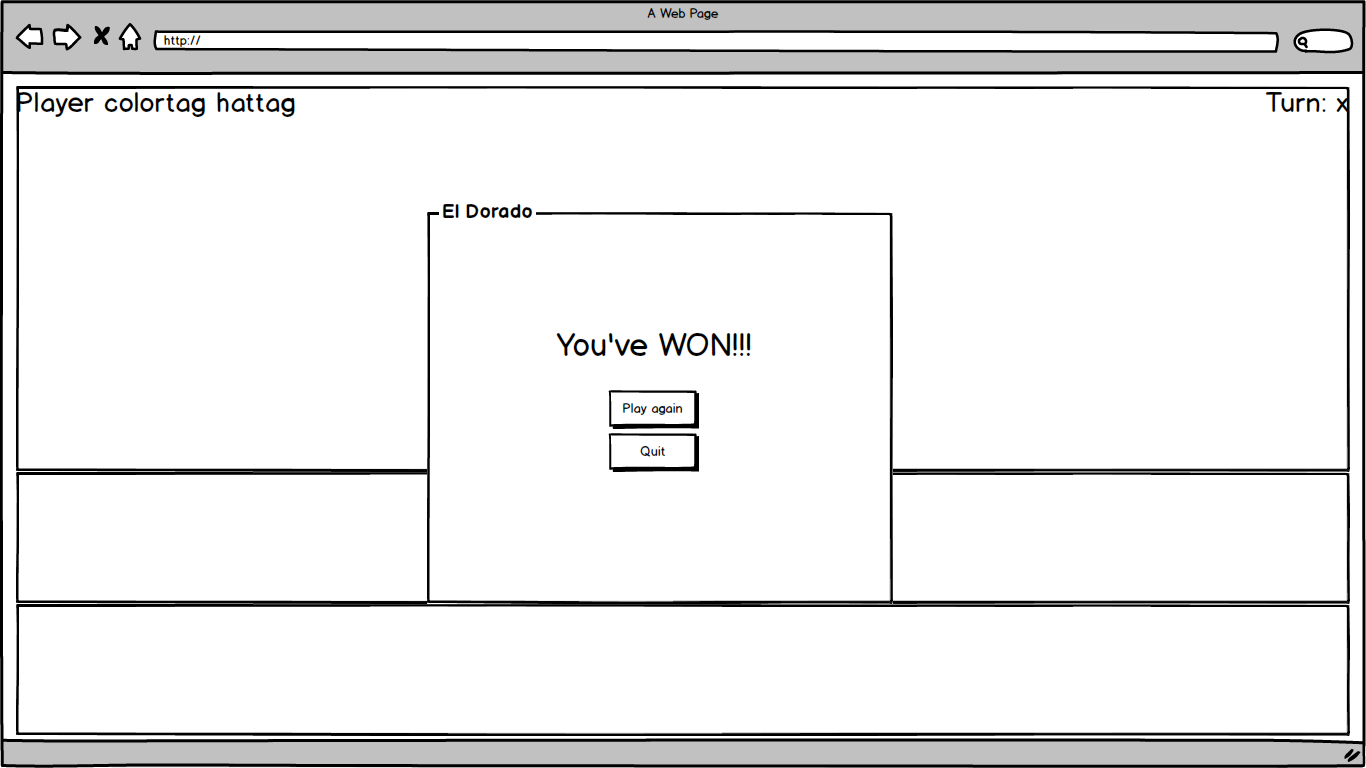


*Figure 10: Market board*



*Figure 11: Stock*

At the end of the game there will be a pop-up to inform all the players of who the winner is. ‘Name’ has won!” will appear on loosing players screen and “You’ve won” on the winning players screen. There will be the option of playing again or to quit. Shown in Figure 12 is the screen of a winning player.



*Figure 12: End of the game*

# Risk Analysis

Following are the identified risks for our Project. Each risk listed in the table has a severity (1 = negligible — 10 = fatal) and an estimate of the probability of occurrences. There is an example, a proactive as well as a reactive method for each risk listed below.

|  |  |  |
| --- | --- | --- |
| **Risks** | **Severity** | **Probability of occurrence** |
| Connectivity issues | 10 | 5% |
| Communication issues | 2-6, depending on the subject of communication | 90% |
| Hardware malfunction | 8 | 5% |
| Time miscalculation | 4 | 75% |
| Scheduling problems | 3 | 80% |
| Estimation problems | 2 | 90% |
| Motivation issues | 1 | 100% |
| Personal / human problems | 7 | 15% |
| Dependencies (one task needs another to be finished) | 7 | 35% |
| Overworking | 4 | 30% |
| Overcomplexity | 4 | 25% |
| Inaccurate requirements | 7 | 10% |
| Requirements misinterpretation | 6 | 15% |

**Connectivity issues:**

Example: The player cannot connect to the server

Proactive method: Choosing a reliable server provider

Reactive method: Switching to a different server

**Communication issues:**

Example: Misunderstanding about a meeting, task, split-up, etc.

Proactive method: Open, easy, straight-forward communication

Reactive method: Resolving issue as soon as possible, making sure that it does not turn into a personal problem

**Hardware malfunction**:

Example: Spilling something over your computer

Proactive method: Treating your electronic devices with care

Reactive method: Getting it fixed as soon as possible, using a different computer in the meantime (Uni/friend/etc.)

**Time miscalculation:**

Example: One task takes too long (can lead to a missed deadline)

Proactive method: Being rather generous with the estimation of needed time

Reactive method: Involve more people from within the group who were not originally assigned this task (if possible), prioritise

**Scheduling problems:**

Example: Cannot find a date where all/most team members are available

Proactive method: Setting fixed dates/times slots aside at the beginning of project

Reactive method: Meetings in smaller groups, working with Skype/Discord/etc.

**Estimation problems:**

Example: Wrongly estimate the size of tasks when splitting them up

Proactive method: Trying to divide the tasks as fairly and equally as possible

Reactive method: Reevaluating the pieces, reassigning the people, helping the person with the biggest task

**Motivation issues:**

Example: Hitting a blockade, being fed up with the project

Proactive method: Keeping the tasks diverse, doing something else in-between

Reactive method: Taking a day off, clearing your head, restart, motivating each other

**Personal / human problems:**

Example: Arguing within the team

Proactive method: Being a nice, well-behaved person, acting like an adult

Reactive method: Solving conflict within team, staying civil and well-mannered

**Dependency issues:**

Example: One task needs to be finished in order to start another one

Proactive method: Good planning, leaving enough room for mishaps (see time miscalculation)

Reactive method: “All hands on deck” for the first task, so it is finished and the depending task can be executed

**Overworking:**

Example: Too much to do in one week

Proactive method: Trying to keep work as evenly distributed as possible (other courses etc.), working together as team to support people when they have other deadlines, exams, etc.

Reactive method: Take a day off, doing something you like (sports, etc.)

**Overcomplexity:**

Example: System is complex/confusing that the user does not understand it/cannot operate it

Proactive method: Testing the system with non-coders/regular people

Reactive method: Getting feedback and reconstructing/ updating for clarity

**Inaccurate requirements:**

Example: The requirements/ user stories do not match the project description

Proactive method: Due diligence when writing the requirements

Reactive method: Rethink the user stories, change requirements (and code if already written)

**Requirements misinterpretation:**

Example: The system does not match the requirements/ user stories

Proactive method: Adhering to requirements when writing code

Reactive method: Change code accordingly

# Project planning until Milestone 2

Following is an outline of the project plan for the coming weeks, with a focus on the upcoming tasks up until milestone 2. Some elements for milestone 3 are already included as well. The main tasks are built on the user stories. The tasks are planned in 7-day interval SCRUM sprints with kickoff Fridays or Saturdays, after the TA-meeting. The tasks are split up in back-end and front-end. Each task has one or more people assigned. The plan is subject to change.

The key responsibilities for this Project, such as choosing the suitable design patterns, creating a User Interface, writing a REST API specification, Implementing the code etc. were divided into smaller tasks, which could be done in a SCRUM Sprint. Every Task is based on a User Story and is sorted by relevance to the key responsibilities, which must be done by a given time. Priorization is given in predicates “must”, “should” and “could” The tasks and the plan do not cover the whole project yet. The Gantt-Chart of these tasks is given in Figure 13.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Priorization** | **Task Name** | **Start Date** | **End Date** | **Duration** |
| Must | Creating User Stories: Robert, Laura, Julia | 23.02.2018 | 28.02.2018 | **5** |
| Must | Risiko Analysis: Alena, Abirami | 28.02.2018 | 02.03.2018 | **2** |
| Must | Creating a User Interface: Robert | 28.02.2018 | 02.03.2018 | **2** |
| Must | Creating a Domain Model: Laura, Julia | 03.03.2018 | 09.03.2018 | **6** |
| Must | Creating a Plan based on the User Stories: Alena, Abirami | 03.03.2018 | 09.03.2018 | **6** |
| Must | Creating an UI Mockup: Robert, Alena | 03.03.2018 | 09.03.2018 | **6** |
| Must | Creating a Class Diagram: Laura, Julia, Abirami | 10.03.2018 | 16.03.2018 | **6** |
| Must | Specification REST API: all | 17.03.2018 | 23.03.2018 | **6** |
| **Should** | **Implementing the cards class: Laura, Julia, Abirami** | 24.03.2018 | 30.03.2018 | **6** |
| Could | Creating and implementing the Action cards |  |  |  |
| Could | Creating and implementing the Joker cards |  |  |  |
| Should | Creating and implementing the Expedition cards |  |  |  |
| **Should** | **Implementing the Board Class: Laura, Julia, Abirami** | 31.03.2018 | 06.04.2018 | **6** |
| Should | Creating and implementing the different fields (incl. networking) |  |  |  |
| Could | Creating and implementing the blockades |  |  |  |
| Could | Creating and implementing the different boards |  |  |  |
| **Should** | **Implementing the Player Class: Laura, Julia, Abirami** | 07.04.2018 | 13.04.2018 | **6** |
| Should | Creating and Implementing the different piles |  |  |  |
| Should | Creating and implementing the playing pieces |  |  |  |
| Could | Creating and implementing the different actions |  |  |  |
| **Should** | **Implementing the Game class: Laura, Julia, Abirami** | 07.04.2018 | 13.04.2018 | **6** |
| Could | Creating and implementing the history function |  |  |  |
| Should | Creating and Implementing Start/ End Game |  |  |  |
| **Should** | **Implementing a graphic User Interface: Robert, Alena** | 24.03.2018 | 13.04.2018 | **20** |
| Could | Creating and implementing the game Interface (until Game Start) |  |  |  |
| Should | Creating and implementing the main board Interface |  |  |  |
| Should | Creating and Implementing the Marketplace Interface |  |  |  |



*Figure 13: Project in a Gantt-Chart*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***Mapping*** | ***Method*** | ***Parameter*** | ***Status Code*** | ***Returned Value*** | ***Description*** |
| */player* | *GET* | *-* | *200* | *List (for all users) ID<integer>, name<string>, username<string>, game <string>* | *List of all players* |
| */player* | *POST* | *name <string>, username <string>* | *201* | *url <string>* | *Add a new player with name and username* |
| */player/details/{playerID}* | *GET* | *playerID <integer>* | *200* | *ID<integer>, name<string>, username<string>, game <string>* | *Show player with playerID* |
| *login* |  |  |  |  |  |
| *logout* |  |  |  |  |  |
| */game* | *PUT* | *players <integer>, difficulty <string>* | *201* | *-* | *Create a new game* |
| */game/{gameID}* | *GET/POST* | *gameID <integer>* | *200* | *url <string>* | *Join an existing game* |
| */player/{playerID}/{hand}* | *GET* | *playerID <integer>* | *200* | *List <Card>* | *List of all cards in hand of player with playerID* |
| */player/{playerID}/{collected\_blockades}* | *GET* | *playerID <integer>* | *200* | *List blockade.value <integer>* | *List of all the blockades collected by player with playerID* |
| */move/moveCard/Select* | *POST* | *card <List Element/tuple>* | *200* |  | *Move card to selection* |
| */move/moveCard/Select/{joker}* | *POST* | *Card <List Element>, Colour <integer>* | *200* | *?? (pop-Up)* | *Move a joker card to selection (???)* |
| */move/moveCard/Select/{action}* | *POST* | *<>* | *200* | *??* | *Move an action card to selection* |
| */move/moveCard/Discard* | *POST* | *card <List element>* | *200* | *-* | *Move card to discard pile* |
| */move/moveCard/Sacrifice* | *POST* | *card <List Element>* | *200* | *(pop-Up)* | *Move card to graveyard* |
| *draw ???* | *POST* |  |  |  |  |
| */move/movePiece* | *POST* | *Field <int>* |  | *Field <int>* | *Move playing piece* |
| */move/movePiece* | *POST* | *Field <int>* | *200* | *???* | *Remove a blockade* |
| */market* | *GET* | *-* | *200* | *List <Cards>* | *Get a list of all the cards available in the marketplace* |
| */stock* | *GET* | *-* | *200* | *List <Cards>* | *Get a list of all the cards in the stock* |
| */market/{card}* | *POST* | *Card <List element>* | *200* | *Position <>* | *Buy a card* |
|  | *POST* |  |  |  |  |
| */history* | *GET* | *-* | *200* | *List <Moves> (??)* | *Show game history* |
| */history/{event}* | *POST* | *event <???>* | *201* | *Event <>* | *Add a move to the history* |
| */history/{move}* | *DELETE* | *-* | *200* |  | *Delete/Undo the last move* |
|  |  |  |  |  | *Switch between playing pieces* |
|  |  |  |  |  | *End turn* |
|  |  |  |  |  | *Exit game* |
|  |  |  |  |  |  |