MoPad

Minimum Requirements Specification

Project	MoPad - A mobile gamepad for browser-based games
Team	Janina Trost (s0541227) Jonas Hartweg (s0541124) Sebastian Alfers (s0541025)
Lecture	Hochschule für Technik und Wirtschaft (Department 4) Internationale Medieninformatik (Master) Web Development Project
Examiner	Dr. Karl Beecher

Table of Contents

1 Introduction

1.1 Motivation and overall aim

Mobile computing is one of the fastest growing markets in the technology landscape. Smartphones are the successful successor of mobile phones and touchscreen enabled devices like tablets pave the way for the future of laptops and desktop computers.
The fundamental idea of this project is to utilize the benefits of touchscreen devices to rethink the way we interact with web based games. This is done with the help of web applications running on mobile devices. This way of controlling games with a gamepad is similar to dedicated console games, where each console like Microsoft's Xbox ship with their own dedicated hardware controller. The smartphone approach to controllers brings benefits in comparison to the hardware solutions.
Firstly, the touchscreen interface gives a lot of flexibility. It adapts according to the game requirements. Custom buttons can be implemented that target game specific controlling scenarios and the interface design can be integrated with the corporate design of the game's interface. Secondly, most smartphones ship with a variety of different sensors like inbuilt accelerometers, which can be utilized as well. Finally, app-based controlling solutions can be distributed via app stores. This brings new revenue sources to game vendors and helps them to monetize their games in a digital world, where pirating software is a common practice.
To target the variety of different mobile operating systems, the controlling application is implemented with web technologies like HTML5, CSS3, WebSockets and JavaScript. It can run in any mobile browser that supports these technologies and additionally, can be distributed via app stores in a wrapping native application with an integrated web view.

1.2 Scope

The MoPad software system allows users to play a browser game and to control it with a controller application on their mobile devices, mainly smartphones. The bridge functions as a mediator between these components.
The *controller*, a website running in a mobile browser, connects with a browser game. It can have various gamepad views with different interfaces. Multiple controllers are supported. The *bridge* is the connector between browser games and the multiple controllers. It controls the registration of games and controllers, the WebSocket connection and the event flow.
The backend offers an API which can be used by any registered game vendor. It offers interfaces to request, register and communicate with controllers. To outline the capabilities of the system, a few demos will be developed that take advantage of the API and its various controlling scenarios. This collection of games is managed in the *GameCenter*.

1.3 Limitations

The proper functioning of the controller requires a stable and fast internet connection. Sending information from the controller to the game and vise versa must happen in a way that the users doesn’t recognize any delay. Mobile data connections might not be stable enough to keep the connection up and a long round trip time might result in lagging inputs.

HTML5 WebSocket connections can’t, due to its W3C specification be established between browser instances. In this case this would be a direct connection between the controller and a game instance. However, since this is not possible, the *bridge*, which also manages the association of a controller with a game instance, takes care of redirecting input events.

The system is targeted at WebKit-based browsers (URL: www.webkit.org). This accounts for the desktop part (namely the game demos) as well as the mobile controller. This covers mainly Google Chrome (desktop and mobile) and Safari (desktop and mobile). It may also run on other browsers.

1.4 Software

Type	Name	URL	Usage
UML modelling	Argo UML	argouml.tigirs.org	Creation of use case diagrams, class diagrams, interface diagrams and flowcharts
Wireframing	Balsamiq Mockups	www.balsamiq.com	Creation of wireframes and mockups
Filesharing	Dropbox	www.dropbox.com	Sharing and synchronization of all files with the team members
Version control and bug tracking	GitHub	www.github.com	Version control with git (www.git-scm.org), documentation and issue tracking
Documentation	Google Drive	drive.google.com	Creation of presentations, text documents and spreadsheets

2 Documentation

2.1 Terminology

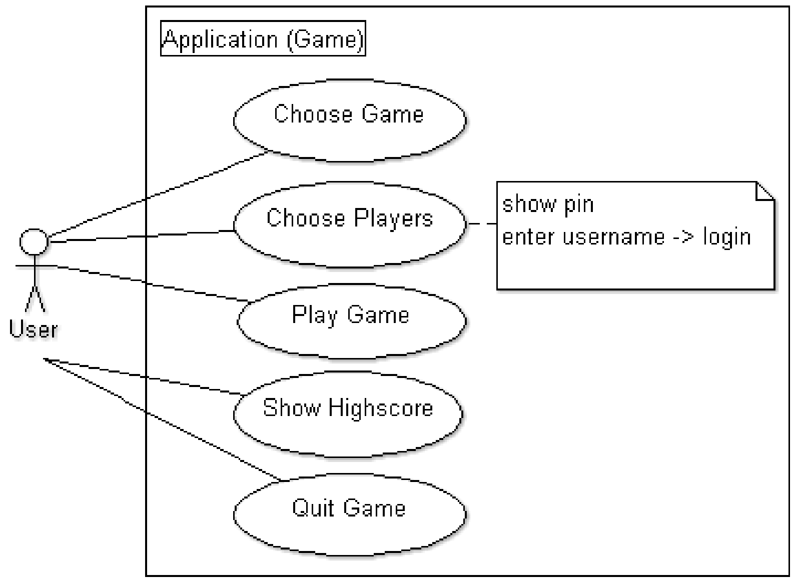
Word	Description
User	A person who uses an app with a mouse and keyboard and not a controller.
Player	A user of a game. He uses a controller and not a keyboard or mouse.
GameCenter	A set of demo games to outline the system's features.
Game	A game that utilizes the controlling options of the system.
Vendor	The producer of the game. A vendor can have multiple games.
Game Instance	A running instance of a *Game* in a browser. It's unique and can only exist one time. After a reload or quit the browser that contains this instance, it does not exist anymore.
Controller	An app that runs in a mobile device like iPhone or iPad to control a *Game Instance.*
Bridge	A web server that connects *Controllers* and *Games.*
View	A view is what the *User* or *Player* interacts with. It can be introduced by a wireframe.
Mockup	Introduces a view and shows the data that is managed by this view. It is an abstract preview and not yet design related.
Pin	Pin is a unique code, generated by the *Bridge,* to associate a *Controller* with a *Game Instance* and connect these.
Gamepad	A Gamepad is a composition of different buttons (InputModuls).

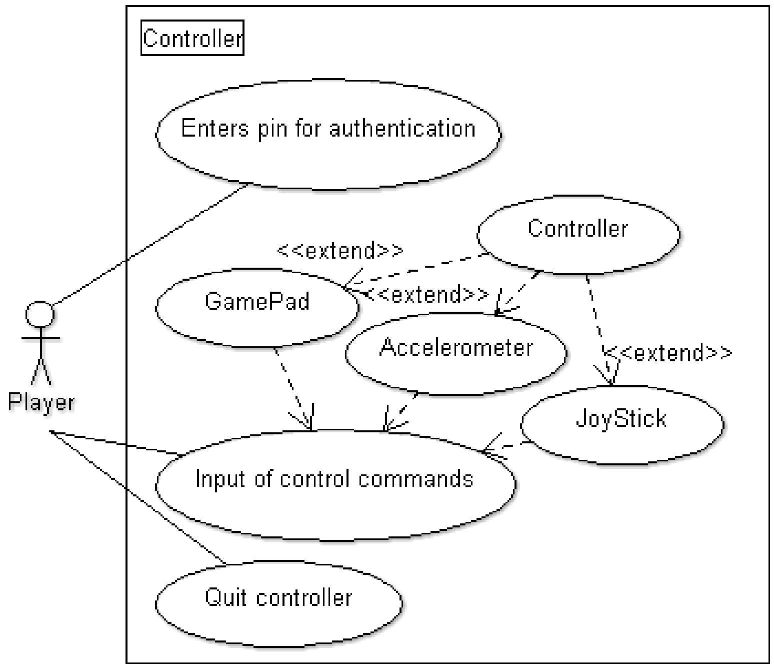
2.2 Use-Cases

The Use-Case-Diagrams give an overview over the domains and interactions of a real-life user with the system.

2.2.1 User and Player

The User takes care of setting up a *Game Instance* with the required *Players*. The *Player* is manipulating the *Game Instance* (aka. playing a game).
Read for more details about “Choose Players” in 2.4.2 Request Controller.

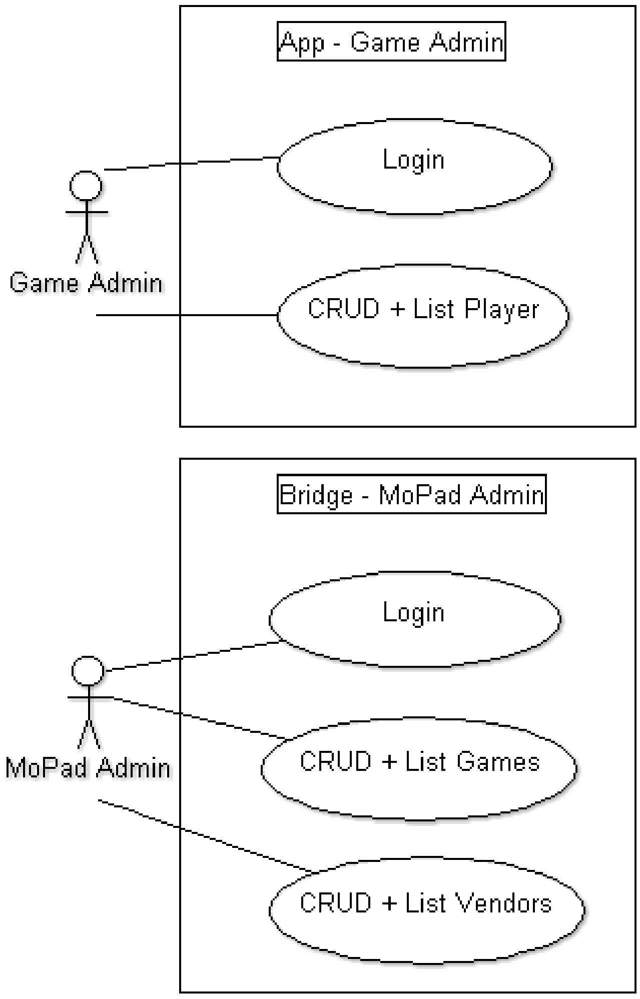




Use-Case-Diagrams: User, Player

2.2.2 Admins

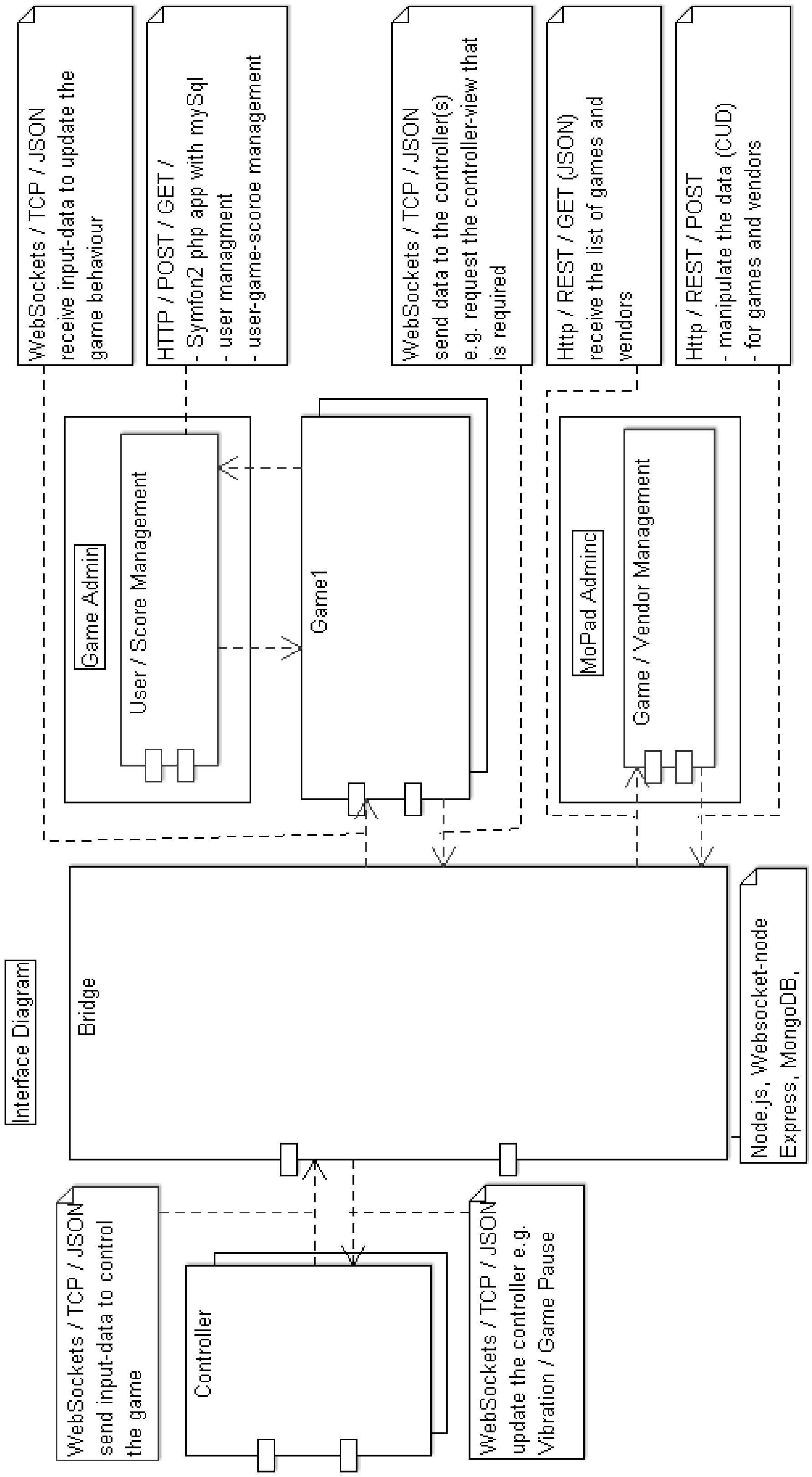
There are two specific use cases that cover data management and administration. The game environment offers a way of reading and deleting *Players* and the associated scores. The administrational interface for the *Bridge* offers a way to add, modify and delete *Game Vendors* and their *Games*.



Use-Case-Diagrams: Admin

2.3 Interfaces

This diagram describes the components of this project and the interfaces between them. It is also a technical overview.



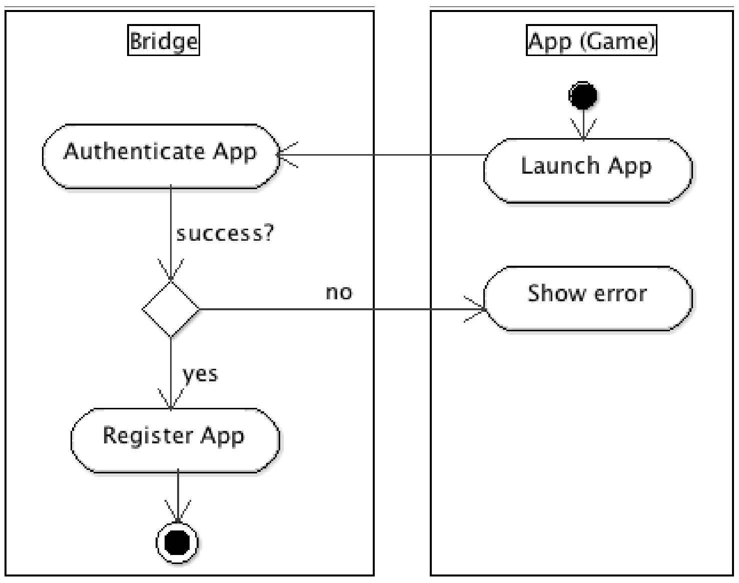
Interface Diagram MoPad

2.4 Flowcharts

A flowchart is a concept for a diagrammatically representation of the steps to a solution or process of achieving a unified goal.

2.4.1 Handshake: Authenticate Game

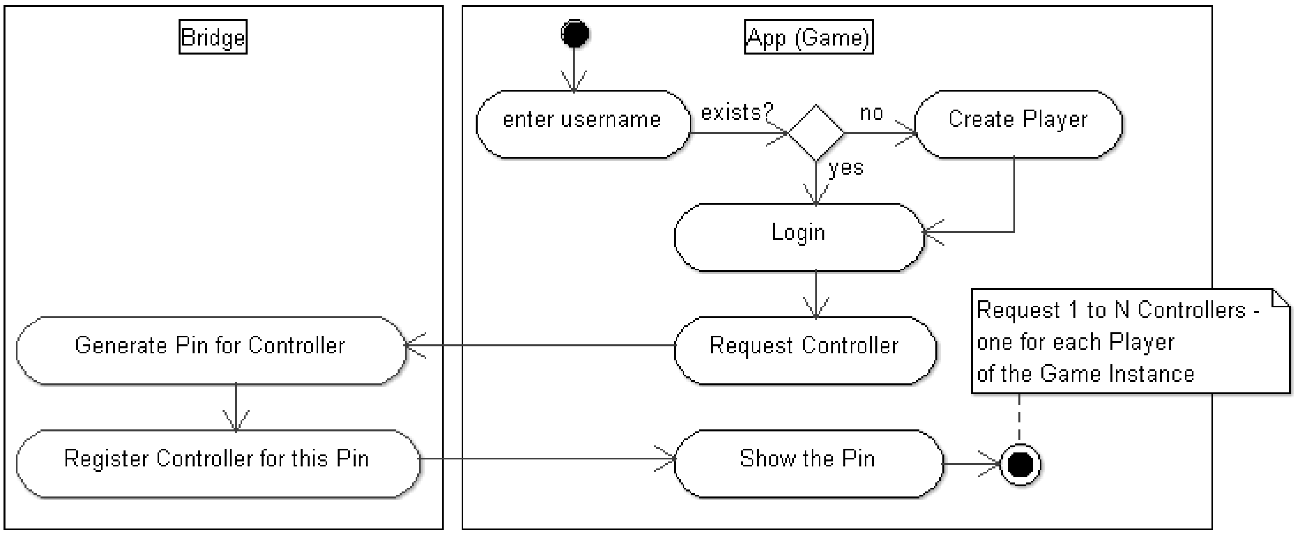
The first step of creating a websocket connection from the *Game Instance* to the *Bridge* is the registration of the WebSocket connection with the registered and verified *Vendors* and *Games* on the *Bridge*. The *Game Instance* identifies itself with a unique access key. It does not include a websocket connection to a controller yet.



Flowchart: Authenticate Game

2.4.2 Handshake: Request Controller

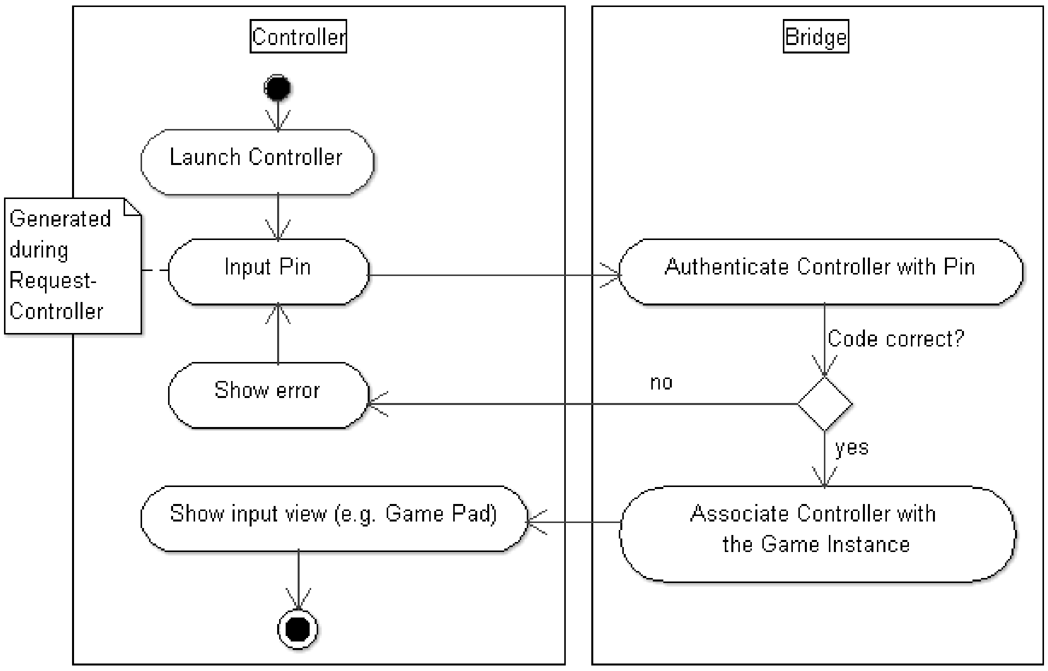
When a verified websocket connection is established between a *Game Instance* and the *Bridge*, a direct connection to a *Controller* can be set up. In this case the input of a username is required to store the users scores for the *Player*. If the entered username doesn't exist, it gets created a new *Player* with the username. The input of password is, for simplification purposes, not required.
When a controller connection is requested by the *Game Instance*, a *Pin* is generated by the *Bridge*. This *Pin* is used to authenticate the *Controller* with the *Bridge* and to establish a unique connection between a *Controller* and a *Game Instance*.


Bridge
App (Game)
enter username
exists?
no
Create Player
yes
Login
Request Controller
Request 1 to N Controllers - one for each Player of the Game Instance
Generate Pin for Controller
Register Controller for this Pin
Show the Pin

Flowchart: Request Controller

2.4.3 Handshake: Authenticate Controller

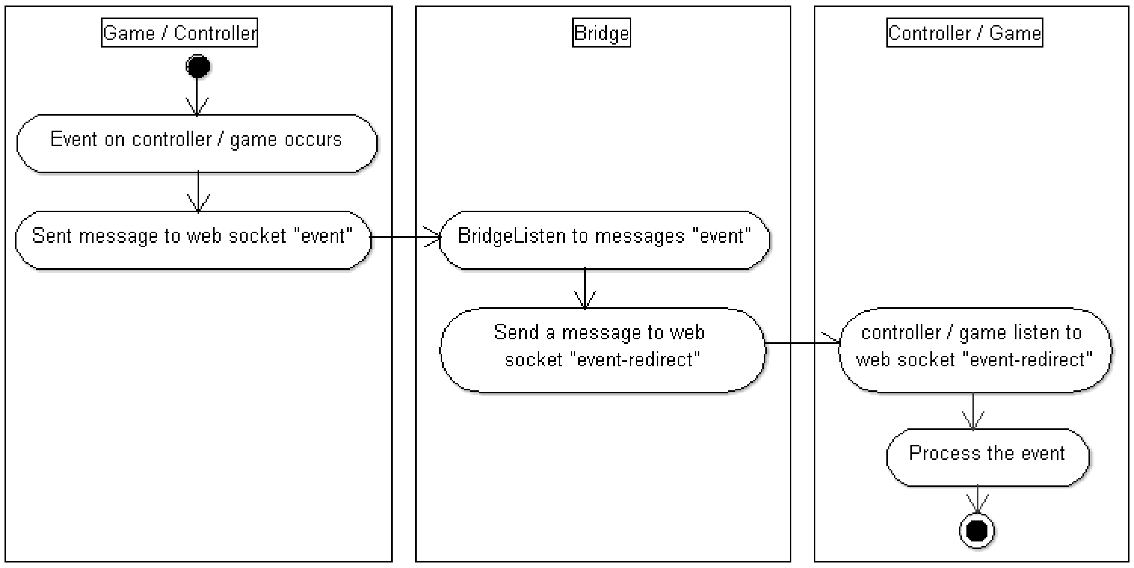
The process of connecting the *Controller* to a *Game Instance* involves entering the previously generated *Pin* on the *Controller*.


Controller
Bridge
Launch Controller
Generated during Request-Controller
Input Pin
Authenticate Controller with Pin
Code correct?
no
Show error
yes
Show input view (e.g. Game Pad)
Associate Controller with the Game Instance

Flowchart: Authenticate Controller

2.4.4 Eventflow

An event is in most cases an input (e.g. button press) on the *Controller*. The *Bridge* only functions as a tunnel to redirect the input to the *Game Instance*.

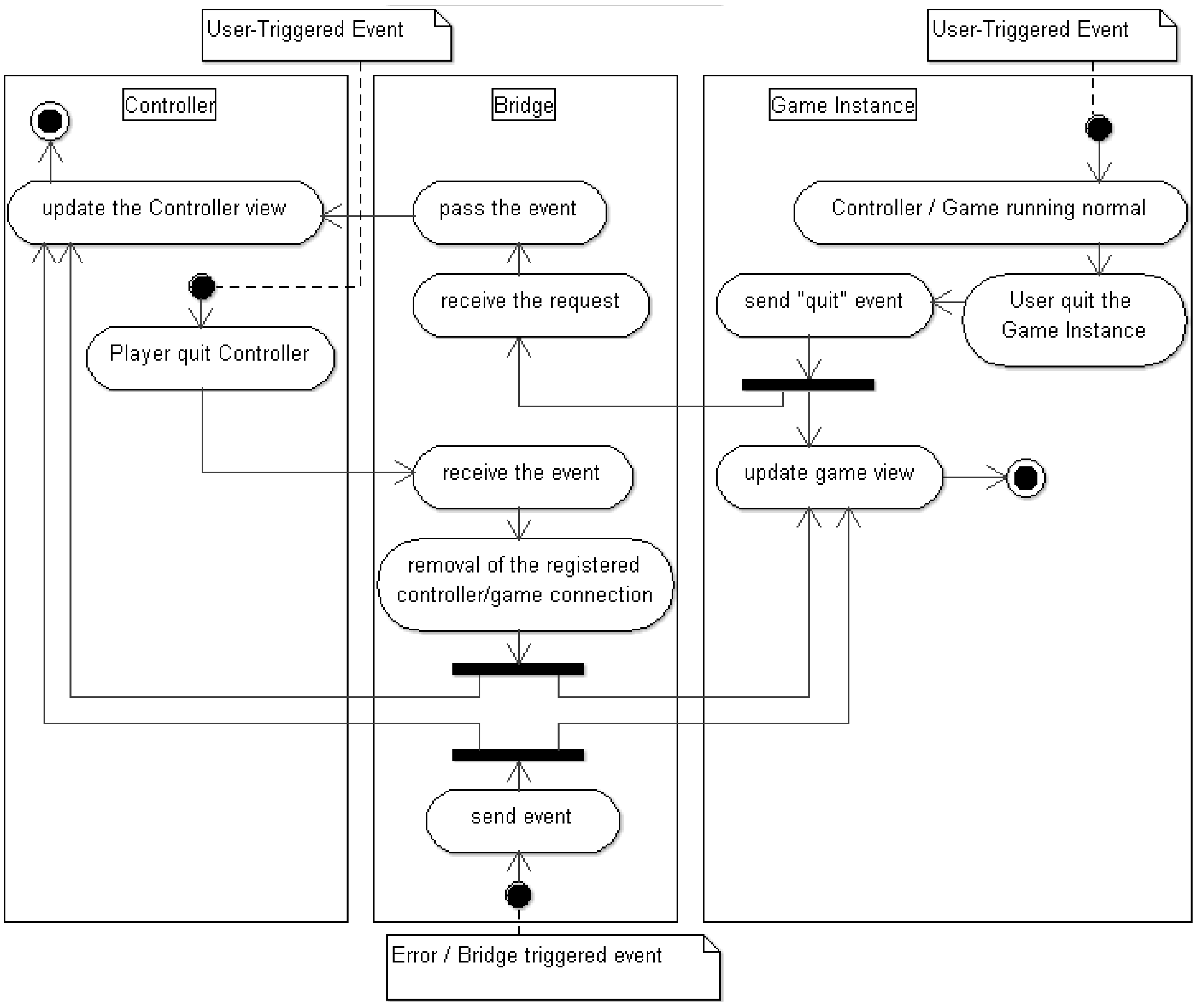


Game / Controller
Bridge
Controller / Game
Event on controller / game occurs
Sent message to web socket "event"
BridgeListen to messages "event"
Send a message to web socket "event-redirect"
controller / game listen to web socket "event-redirect"
Process the event

Flowchart: Events

2.4.5 Quitting the controller or the game

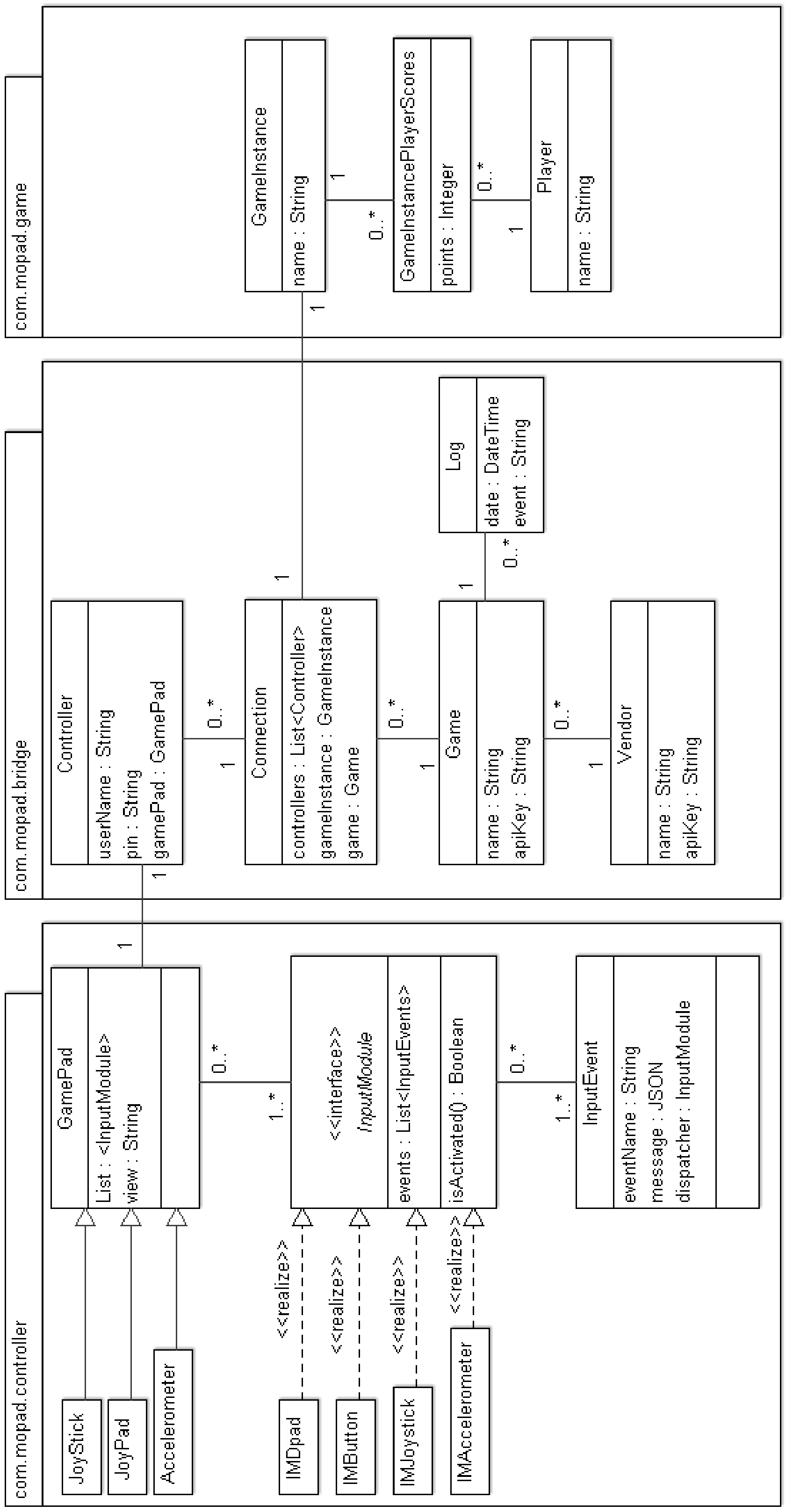
In the case of closing the *Game Instance* or closing the *Controller* results in a loss of the controller connection. This event has to be reported to the *Game Instance* associated *Controller* or *Game Instance* respectively. The connection also has to be deregistered on the *Bridge*.



Flowchart: Quit

2.5 Classes

This diagram gives an overview of the data that is handled on each component and the connection between them. It does not show how the data is connected, if they are not processed on the same component.



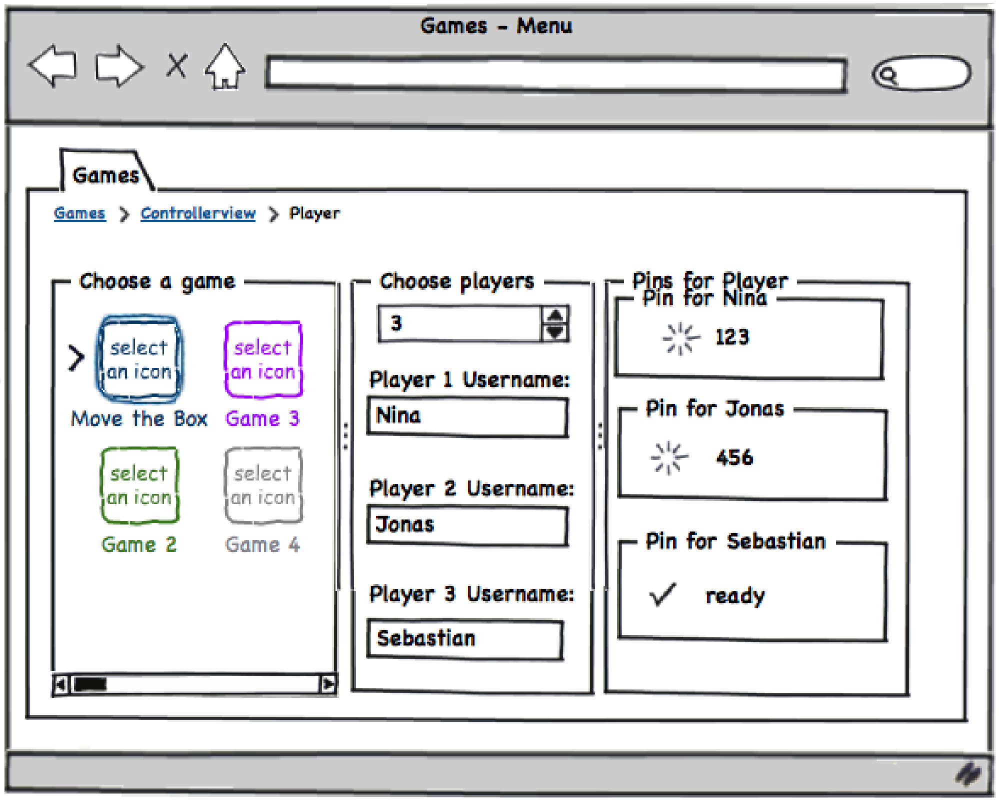
2.6 User interface

These wireframes outline, what data is processed and displayed in each view. It also shows the environment, where each view is displayed (e.g. mobile app or browser).

2.6.1 GameCenter

2.6.1.1 Games Menu

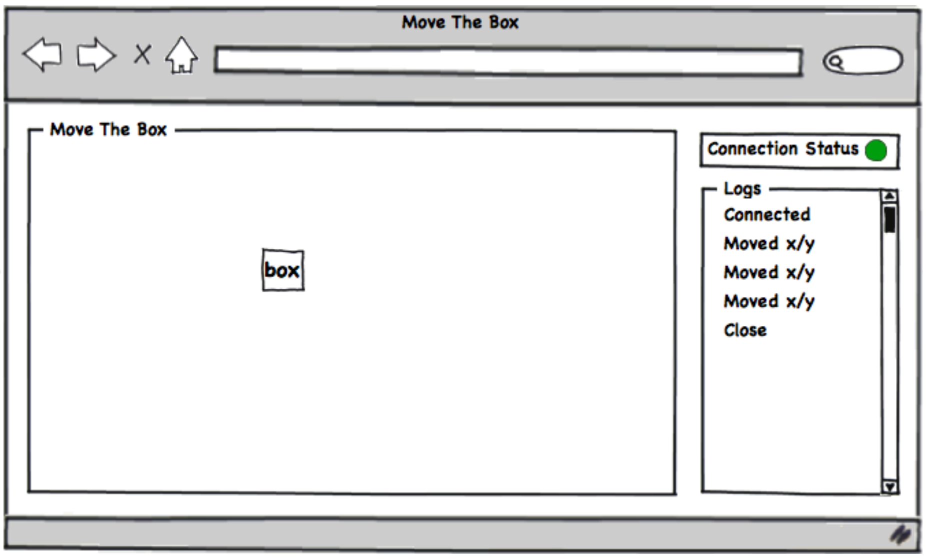
This view is the point where a user creates a *Game Instance* and chooses the *Players* for this game. After a *Player* is authenticated by his username, a *Pin* is generated and displayed for each *Player*.
(Compare 3.2.1 Authenticate App and 3.2.2 Request Controller)



Wireframe: Games Menu

2.6.1.2 Game “Move the Box”

A very simple game that can be manipulated by different kinds of *Controllers*. (Compare 3.6 The Game “Move the Box” and 3.3 Events)

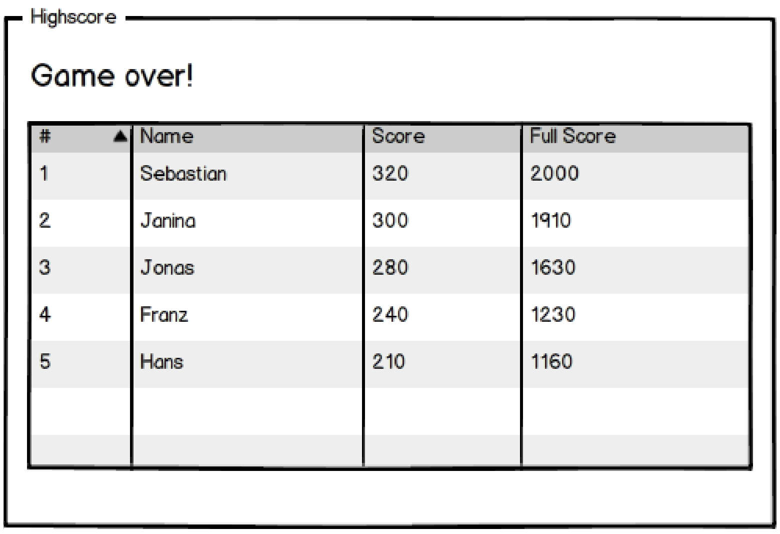


```
Connected
Moved x/y
Moved x/y
Moved x/y
Close
```

Wireframe: Game “Move the box”

2.6.1.3 Message (Example: Highscore)

A view that splits up in a message area and a content area. It can show every kind of information and is mostly game-related. I can be shown in the game or in the controller. (Compare 3.4 Quit Game)



Game over!

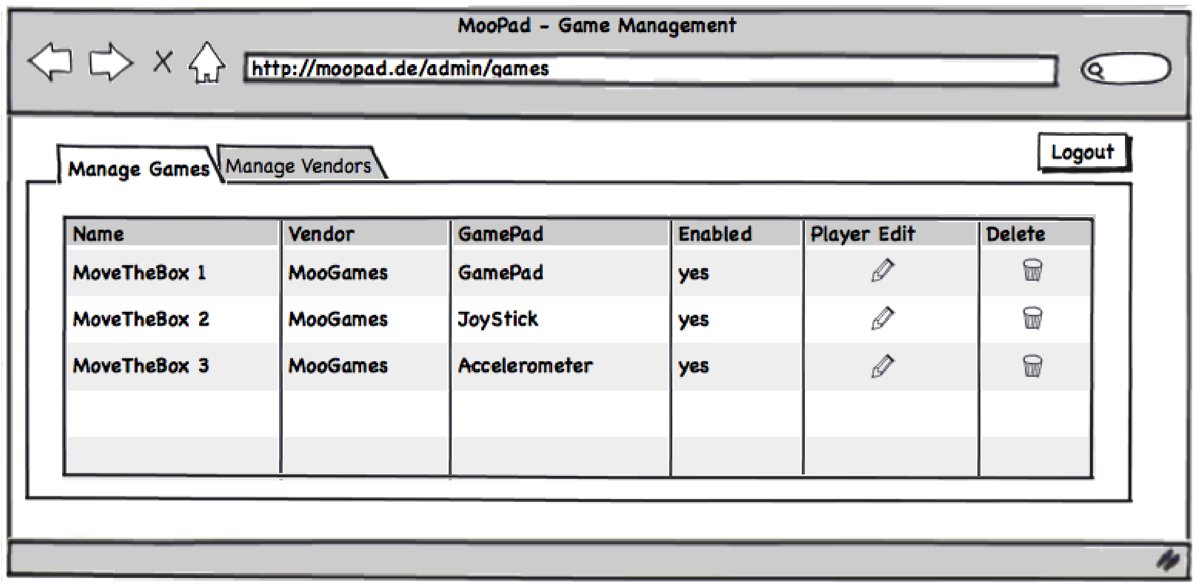
# ▲	Name	Score	Full Score
1	Sebastian	320	2000
2	Janina	300	1910
3	Jonas	280	1630
4	Franz	240	1230
5	Hans	210	1160

Wireframe: Message (Example: Highscore)

2.6.2 Bridge

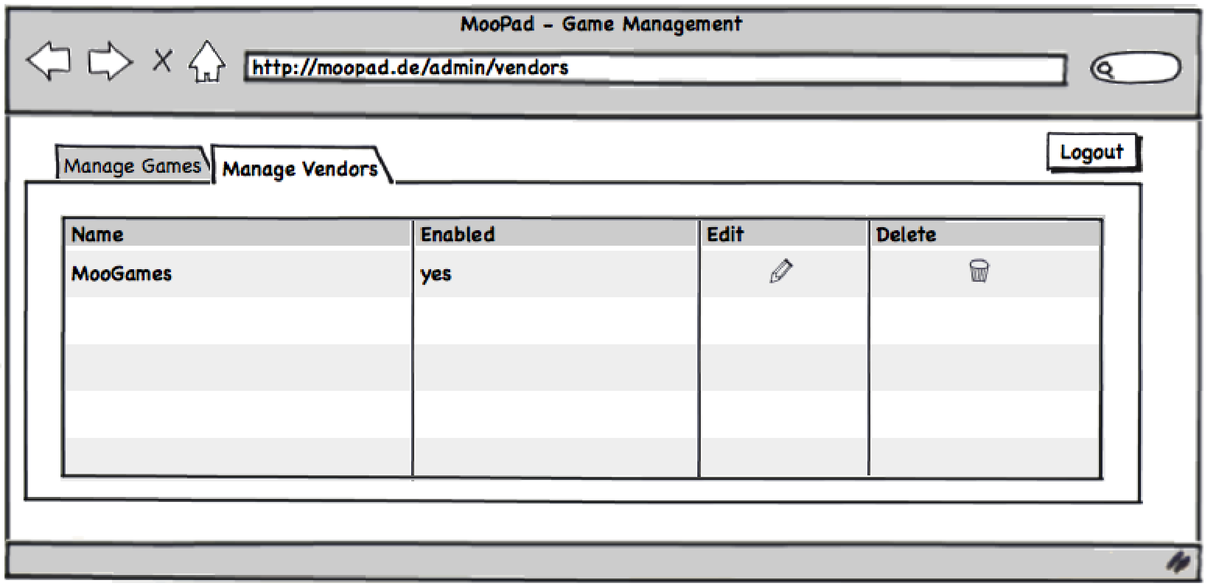
2.6.2.1 Admin Management of Games and Vendor

These are the views for the *MoPad Admin*. They are responsible for managing *Games* and *Vendors*.
(Compare 3.1.1 Game Management and 3.1.2 Vendor Management)



Name	Vendor	GamePad	Enabled	Player Edit	Delete
MoveTheBox 1	MooGames	GamePad	yes	✎	🗑
MoveTheBox 2	MooGames	JoyStick	yes	✎	🗑
MoveTheBox 3	MooGames	Accelerometer	yes	✎	🗑

Wireframe: Game Management

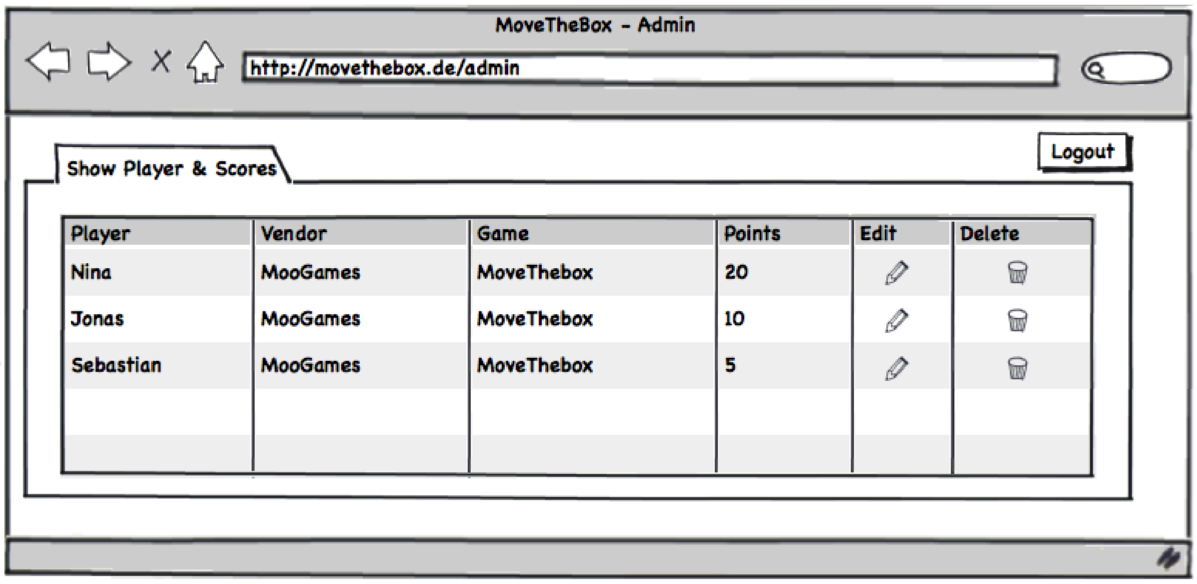


Name	Enabled	Edit	Delete
MooGames	yes	✎	🗑

Wireframe: Vendor Management

2.6.2.2 *Admin Management of Users*

This view is the admin for the *Game*. I shows the *Player*, that are registered. If a *Game* is created and a not registered *Player* is added, he will be registered with his username. If the *Player* is already registered, it will be loaded.
(Compare 3.1.3 User Management)



Player	Vendor	Game	Points	Edit	Delete
Nina	MooGames	MoveThebox	20	✎	🗑
Jonas	MooGames	MoveThebox	10	✎	🗑
Sebastian	MooGames	MoveThebox	5	✎	🗑

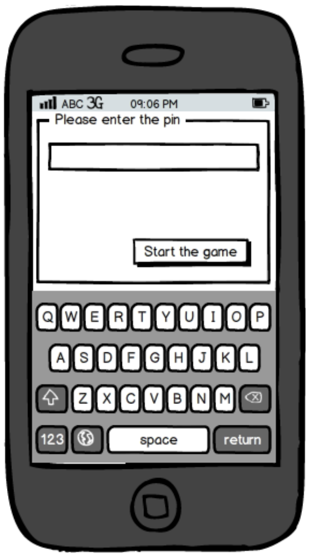
Wireframe: Player and score management

2.6.3 Controller

As shown before, a *Controller* consists of one or more *InputModules* that build the *Gamepad*. Besides the possibility of controlling games, it can also used as a backchannel to display information relevant to the *Game Instance*, like scores.

2.6.3.1 Enter Pin

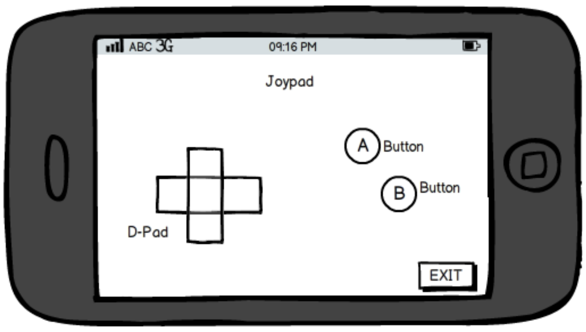
When the controller app is started, this view is show. It gives the *Player* the possibility to connect to a *Game Instance*. The controller view, that is required for the *Game Instance* is selected automatically after entering the *Pin*.
(Compare 3.2.3 Authenticate Controller)



Wireframe: Pin input

2.5.3.2 Gamepad "Joypad"

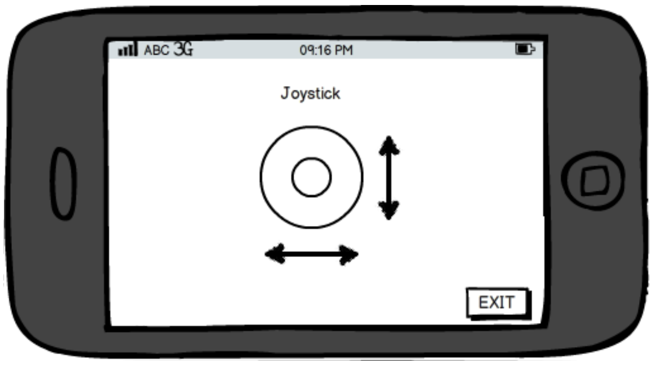
A controller view "Joypad" to control a *Game Instance*. A *Gamepad* is a composition of different buttons, the *InputModuls*. (Compare 3.5.1 Gamepad "Joypad")



Wireframe: Gamepad "Joypad"

2.6.3.3 Gamepad “Joystick”

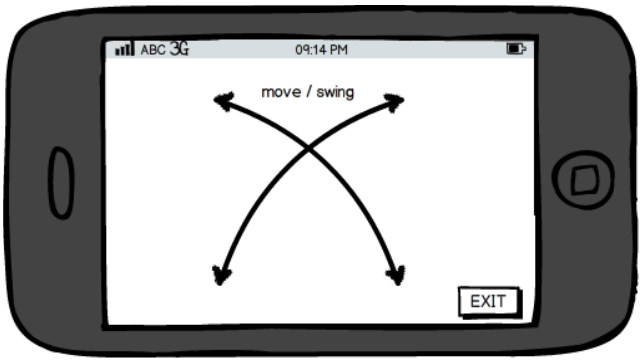
A controller view “Joystick” to control a game instance. It is a button that can be moved in different directions. (Compare 3.5.2 Gamepad “Joystick”)



Wireframe: Gamepad “Joystick”

2.5.3.4 Gamepad “Accelerometer”

A controller view “Accelerometer” to control a game instance. Is makes usage of the accelerometer that is build in some mobile devices. This view has a feature detection for this hardware device and a fallback to the view joystick controller view.
(Compare 3.5.3 Gamepad “Accelerometer”)



Wireframe: Gamepad “Accelerometer”

Appendix (Use-Case specifications)

3.1 Admin

3.1.1 Game Management (MoPad Admin)

Goal:	Manage the registred games on the bridge
Precondition:	The admin is logged into the game and vendor admin interface
Postcondition (success):	The admin is able to manage the games
Postcondition (error):	Error message
Stakeholder:	MoPad Admin
Trigger Event:	A specific maintenance URL is visited
Description:	A game can only connect to the bridge, if it is registered and activated. The game and the credentials are managed here. The game instances use this credentials to authenticate on the bridge. Each game is associated with a game vendor.
Extension:	More settings for security and logging
Alternatives:	-

3.1.2 Vendor Management (MoPad Admin)

Goal:	Manage the games on the bridge
Precondition:	The admin is logged into the game and vendor admin interface
Postcondition (success):	The admin is able to manage the vendors
Postcondition (error):	Error message
Stakeholder:	MoPad Admin
Trigger Event:	A specific maintenance URL is visited
Description:	A game can only connect to the bridge, if it is registered and activated. Each game is associated with a vendor and the vendors credentials are managed here.
Extension:	More settings for security and logging
Alternatives:	-

3.1.3 User Management (GameCenter Admin)

Goal:	Manage the game players
Precondition:	The user is logged into the GameCenter admin
Postcondition (success):	The user is able to manage the users
Postcondition (error):	Error message
Stakeholder:	Game Admin
Trigger Event:	Open the admin area for the game
Description:	List, Edit and Delete users that are registered for the game
Extension:	Extended functionality for the users like password forgotten, disable and notifications
Alternatives:	-

3.2 Game-Bridge-Controller Handshake

3.2.1 Authenticate App

Goal:	Register game on the bridge
Precondition:	User choose a game
Postcondition (success):	App is registered on bridge, User can request a controller and the game instance is created
Postcondition (error):	Error message
Stakeholder:	User
Trigger Event:	User chooses a game
Description:	A game instance is registered on bridge
Extension:	User has to be logged in
Alternatives:	-

3.2.2 Request Controller

Goal:	Generate a pin on the bridge and associate it to a game instance
Precondition:	A game is registered
Postcondition (success):	Display pin in app, user can use pin for controller-registration
Postcondition (error):	Error message
Stakeholder:	User
Trigger Event:	The game to be played was chosen in the app
Description:	The app requests a controller from the bridge. The bridge generates a pin and associated with the game instance. The app displays the pin so that the user can use the pin to register a controller. A pin is associated to a user that is authenticated only by his username.
Extension:	The pin contains encoded informations about the game instance and associated user
Alternatives:	-

3.2.3 Authenticate Controller

Goal:	Register controller on bridge, bridge connect the controller with the game instance
Precondition:	Game instance is registered, a controller is requested and the generated pin is displayed in app. The controller is running on the mobile device
Postcondition (success):	The bridge is connected to the controller with the game instance, display the controller input-view
Postcondition (error):	Error message
Stakeholder:	Player
Trigger Event:	The user requests a controller
Description:	The player enters the displayed pin on the controller. The controller registers itself on bridge. The bridge associates the controller with the game instance.
Extension:	-
Alternatives:	-

3.3 Events

Goal:	Controller send an event (e.g. action-command) to a game instance or the game send an event (e.g. reaction-commands) to controller to process them
Precondition:	The controller is connected with the game instance
Postcondition (success):	The bridge receive the event and forward this without an error, the controller/game process the event
Postcondition (error):	If the bridge lose the connection to controller or game, the bridge send an error-message to the sender from the event, and invite the user to reconnect the controller with the game
Stakeholder:	-
Trigger Event:	An event occurs (e.g. player presses a button or connection is lost / broken)
Description:	An "event" are all actions that send a piece of information from the controller to the game and vice versa
Extension:	-
Alternatives:	-

3.4 Quit Game

Goal:	Notify the Game instance and Controller that the respective participant has lost the connection
Precondition:	The connection from the Controller via the Bridge to the Game instance is stable
Postcondition (success):	Do not send data to the WebSocket (bridge)
Postcondition (error):	-
Stakeholder:	Player and User
Trigger Event:	a) Controller is shut down (aka browser quit / phone shut down) b) Game instance is shut down (aka browser quit / Computer is shut down) c) Bridge losses connection to the pool of web sockets
Description:	Show an icon with red color to identify that the connection is not stable
Extension:	-
Alternatives:	If the device, that runs this controller / game, is not able to run it (aka. an old browser version), it shows an error

3.5 Controller

3.5.1 Gamepad “Joypad”

Goal:	A game instance can be manipulated with an input layout that looks and acts like a gamepad. This runs in a mobile device
Precondition:	The handshake between the Controller and the Game instance must be successful
Postcondition (success):	A Game instance can be controlled
Postcondition (error):	Show status "connection error"
Stakeholder:	Player
Trigger Event:	A game with the controller type "gamepad" was chosen
Description:	On the screen the player see buttons like a nintendo-pamepad: a cross with the keys "right", "left", "up" and "down". Also two Buttons "A" and "B". These buttons can renamed in "start" and "stop" or otherwise.
Extension:	-
Alternatives:	If the device, that runs this controller, is not able to run it (aka. an old browser version), it shows an error

3.5.2 Gamepad “Joystick”

Goal:	A game instance can be manipulated with a controller with an input layout that looks and acts like a "normal" joystick.
Precondition:	The handshake between the controller and the game instance must be successful
Postcondition (success):	A game instance can be controlled
Postcondition (error):	Show status "connection error"
Stakeholder:	Player
Trigger Event:	A game with the controller type "joystick" was chosen
Description:	Moving the "joystick" on the screen of the mobile device. The input is done with the finger (touch gesture)
Extension:	-
Alternatives:	If the device, that runs this controller, is not able to run it (aka. an old browser version), it shows an error

3.5.3 Gamepad “Accelerometer”

Goal:	Gesture-based control of game functions
Precondition:	A game controller that defines the accelerometer as one of its input types
Postcondition (success):	The game instance reacts on the input from this controller
Postcondition (error):	Fall back to the gamepad “joystick”
Stakeholder:	Player
Trigger Event:	A game is launched that uses the input method (controller) "accelerometer"
Description:	Controlling of a game by tilting and rotating the controller (mobile device)
Extension:	-
Alternatives:	The input method "joystick" can substitute an accelerometer, if the controlling device doesn't have an accelerometer

3.6 The Game “Move the Box”

Goal:	Game that allows one or multiple players to move his/their own box on the display
Precondition:	One or multiple controllers are connected with the game instance
Postcondition (success):	The box moves to its desired position
Postcondition (error):	-
Stakeholder:	One or multiple Player
Trigger Event:	The game is launched and player have their controllers connected
Description:	Game that allows one or multiple players to move his/their own box on the display
Extension:	-
Alternatives:	-