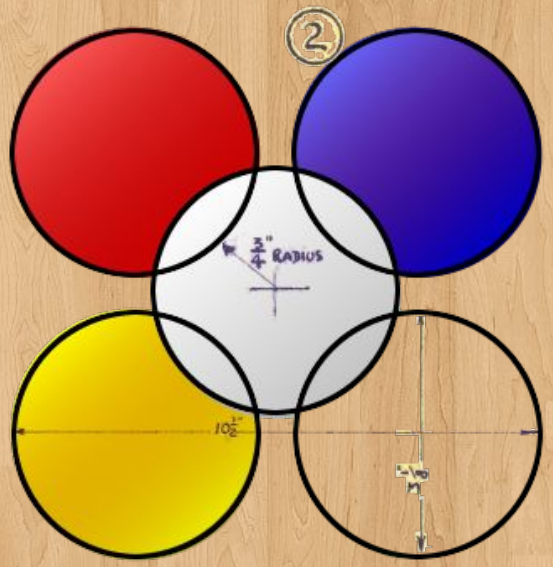


Board Game Arena

STUDIO

Online boardgaming development platform

The BGA framework at a glance

Intro

The BGA framework helps you to build a online board game adaptation:

- That will be played in realtime.
- That will be played by several human players.
- That will be played from an internet browser.
- That will have rules enforcement (ie : no possibility to cheat).

This presentation shows at a glance how BGA framework is working

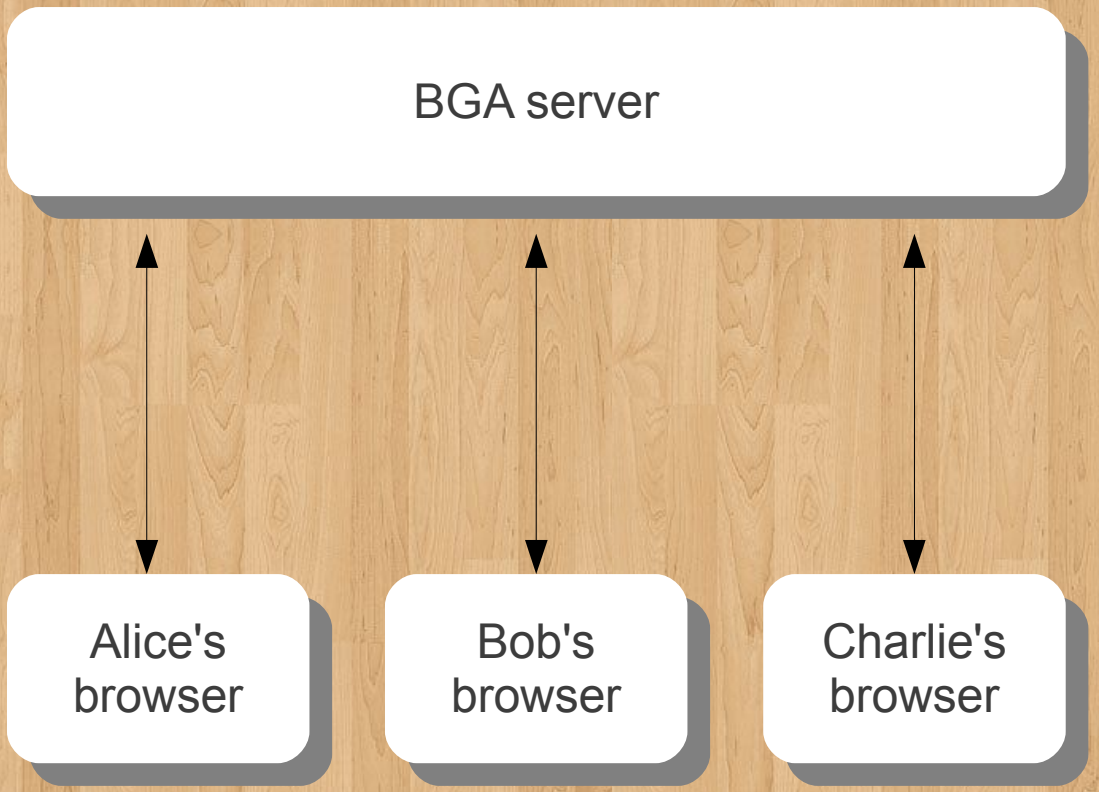
After reading this presentation, you'll be able to understand how a game is running on the BGA platform.
We won't go into details, in order you can have a good overview of the platform, and afterwards immediately understand « which component is doing what ».

Board Game Arena
STUDIO

Clients & server

Here's an overview of the BGA architecture.

3 players : Alice, Bob and Charlie are playing a game on the BGA server using their web browser.

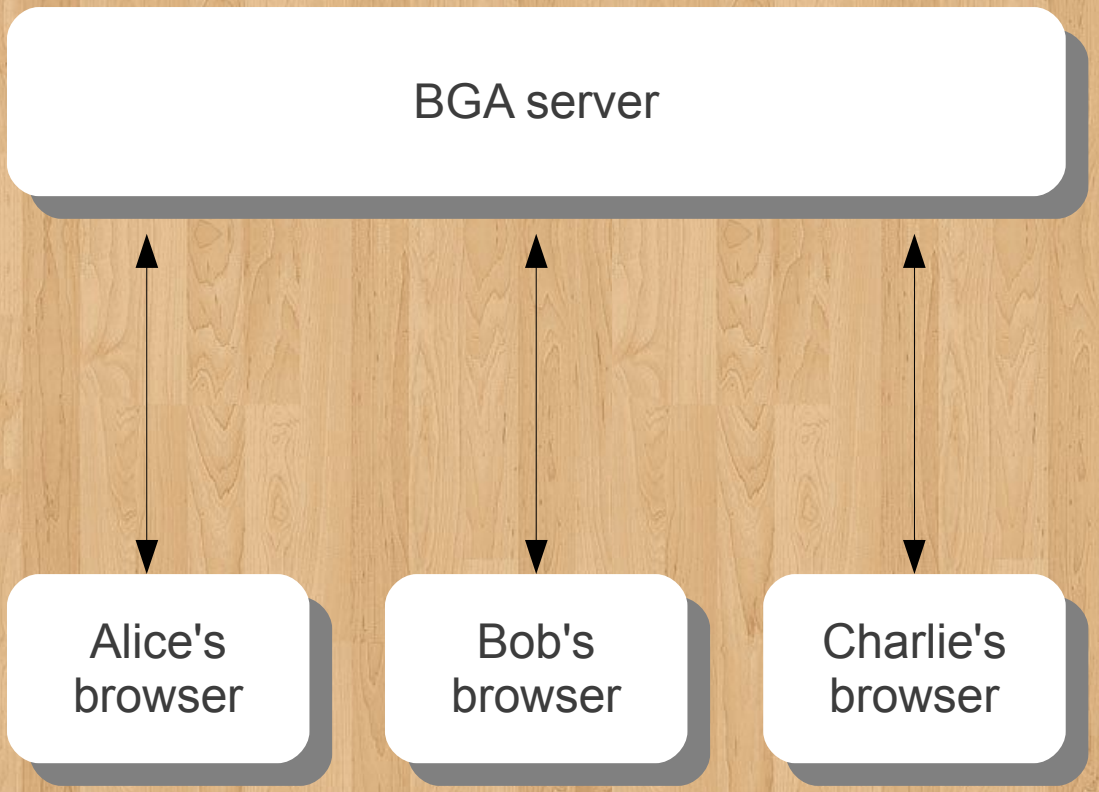


Board Game Arena
STUDIO

Clients & server

While developing your game logic, you'll have to work on both server side and client side.

Let's see what's on each side.



Board Game Arena STUDIO

The server side

On the server side, you'll find 3 components :

• The **game database** :

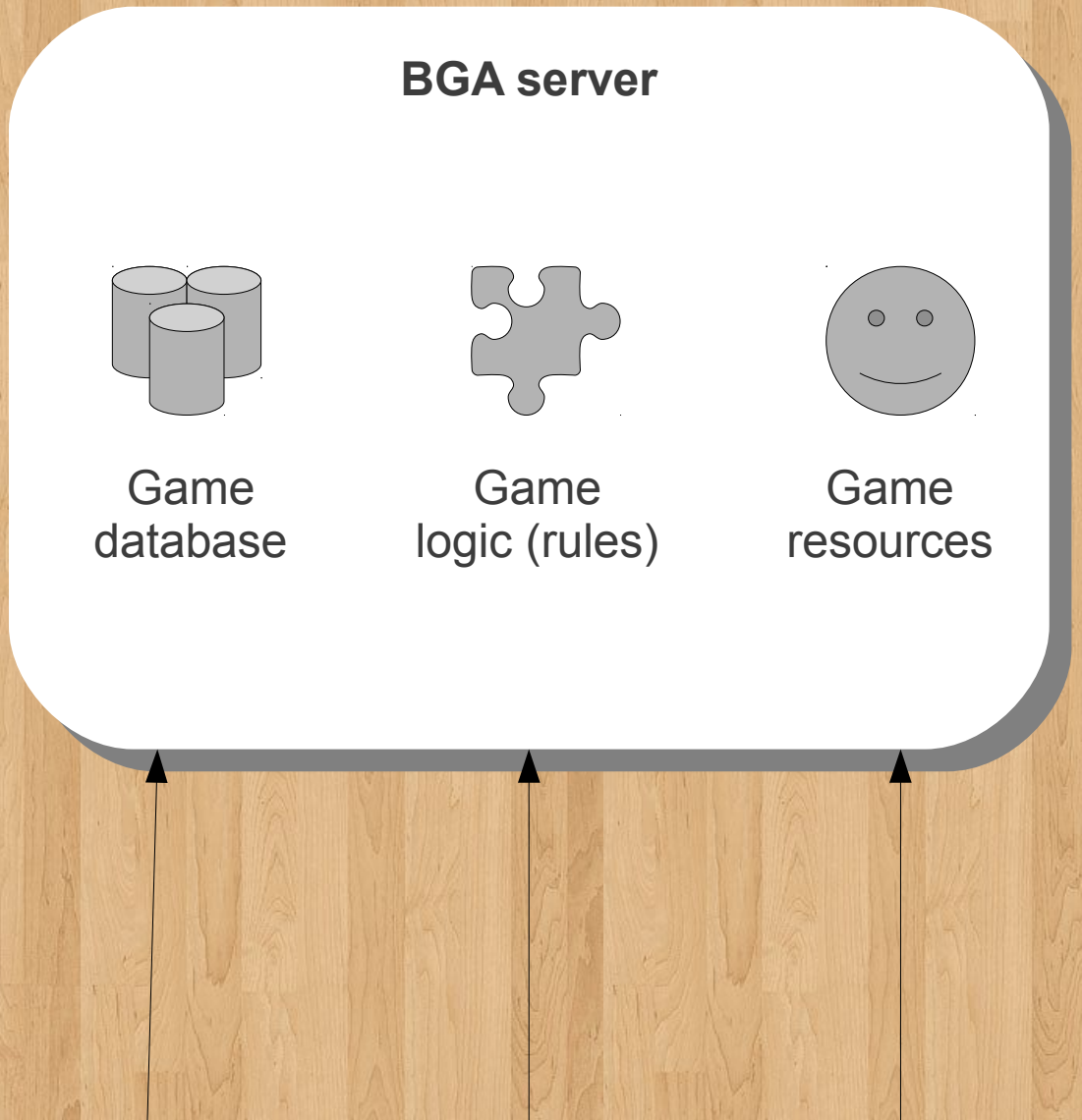
This is the current state of the game (ex : scores, tokens place on board, which card is where...)

• The **game logic** :

This is the rules of the game. The game logic ensures that it's not possible to cheat (« rules enforcement »).

• The **game resources** :

This is what's going to be downloaded to be used by the client browsers (ex : images, stylesheets, ...).

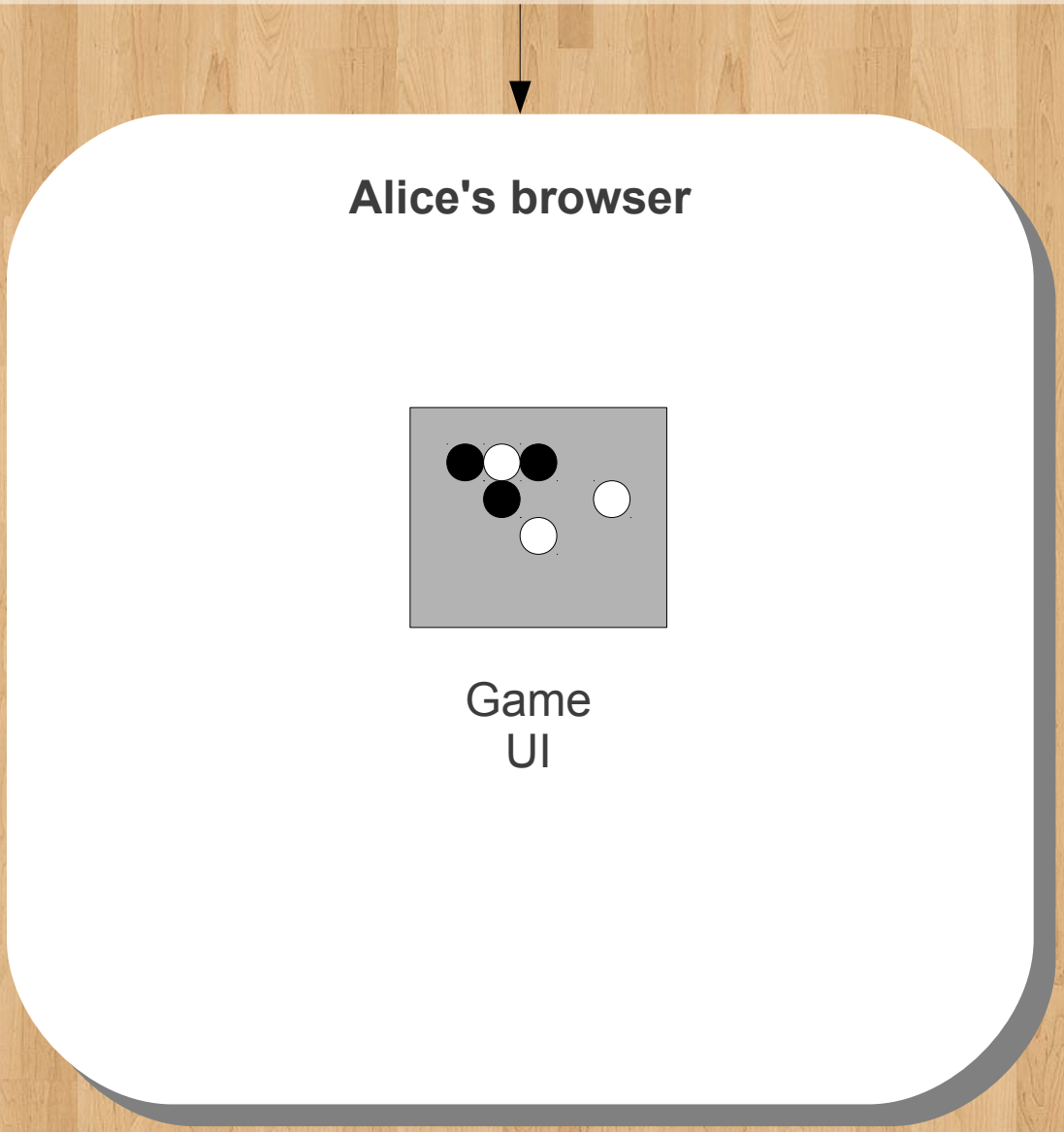


Board Game Arena
STUDIO

The client side

On the client side, you'll find the **user interface (UI).**

The user interface manage all what is displayed to the user (ex : « this token has moved from A to B »), and make possible all user actions on the game (ex : « a click on this card plays this card »).

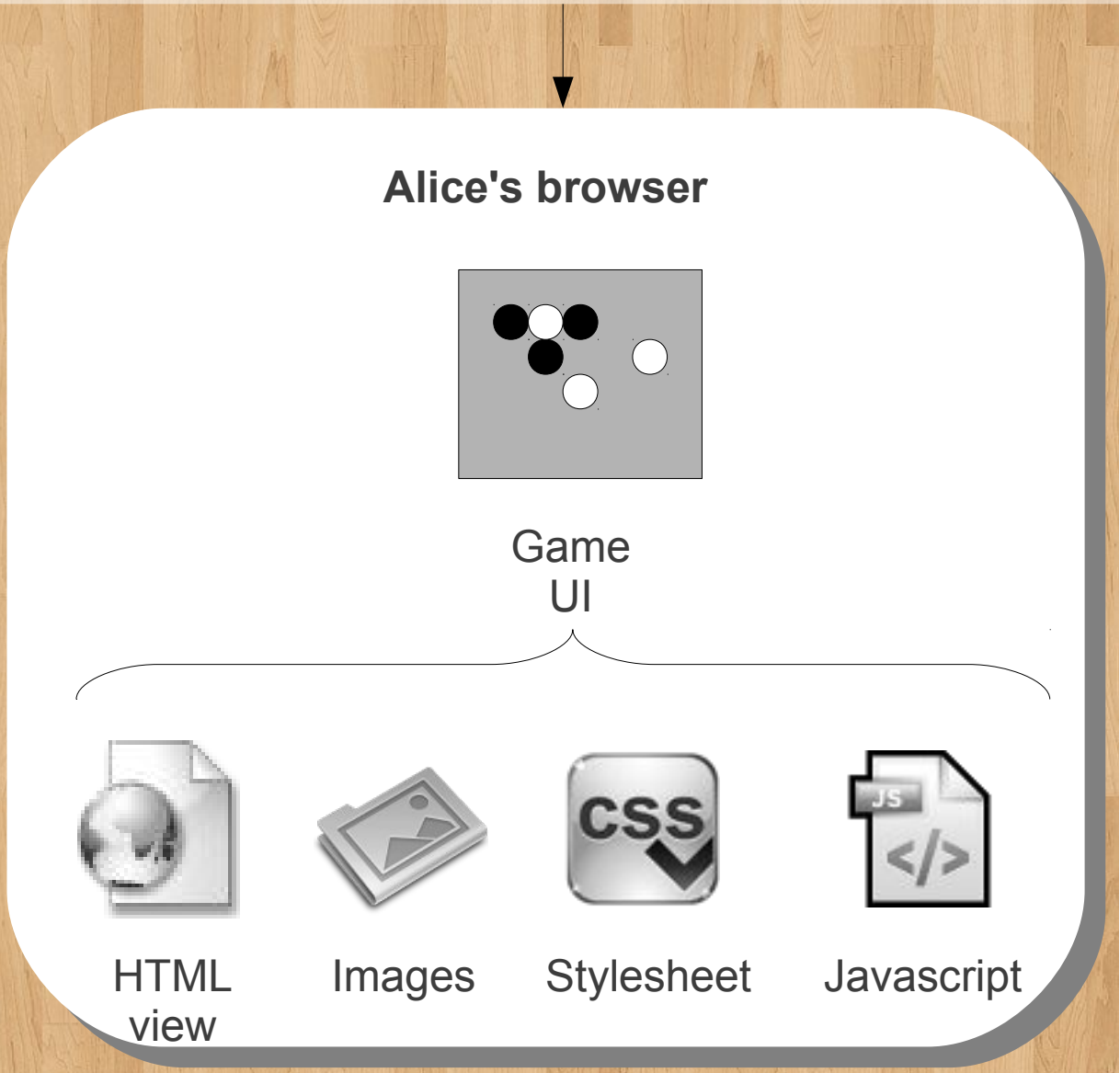


Board Game Arena
STUDIO

The client side

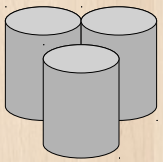
To do this, you will use 4 different types of resources :

- A **HTML view** which defines the basic layout of your game UI.
- Some **images** to display the game art.
- A **CSS stylesheet** which defines the style and position of elements of your game UI.
- A **javascript script** which defines the mechanisms of your UI (ex : click on this button trigger this action).

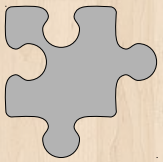


Board Game Arena STUDIO

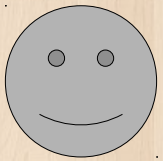
Summary : technologies used



The game database is using **MySQL.**



The game logic is using the **PHP** language**.**



The game ressources, used for the user interface, are using :

HTML language (HTML4)

CSS stylesheet

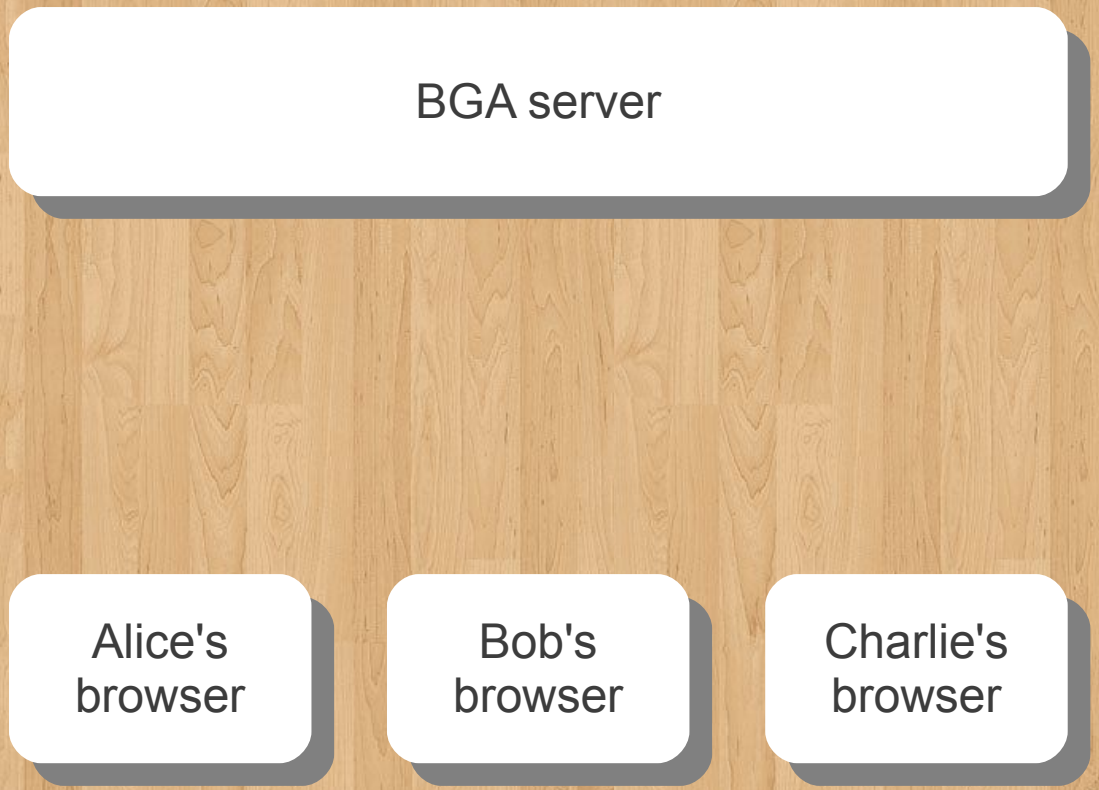
Javascript script (with Dojo framework)

Board Game Arena STUDIO

Information flow

Now, let's have a look on how all these components interact with a simple example.

Our three players A, B and C are starting a game together. The name of this fake game is «mygame ».



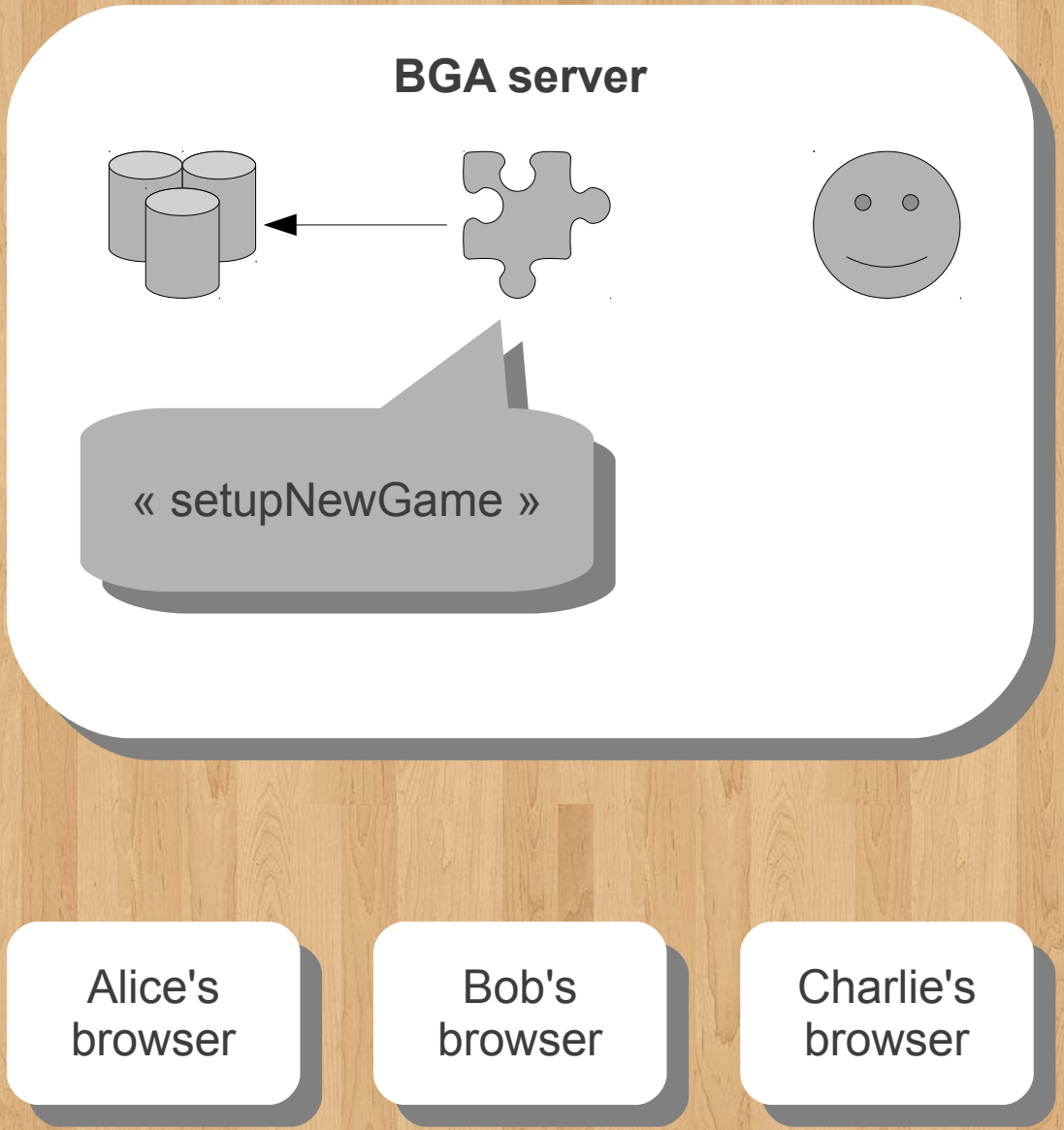
Board Game Arena STUDIO

Creating a new game

As soon as everyone accept to start the game, your PHP method « setupNewGame » is called (from game logic).

This method must setup the initial game situation, as described in game rules, ie you have to create the right SQL statements to initialize the database according to the game rules.

This is it, you can now welcome your players !



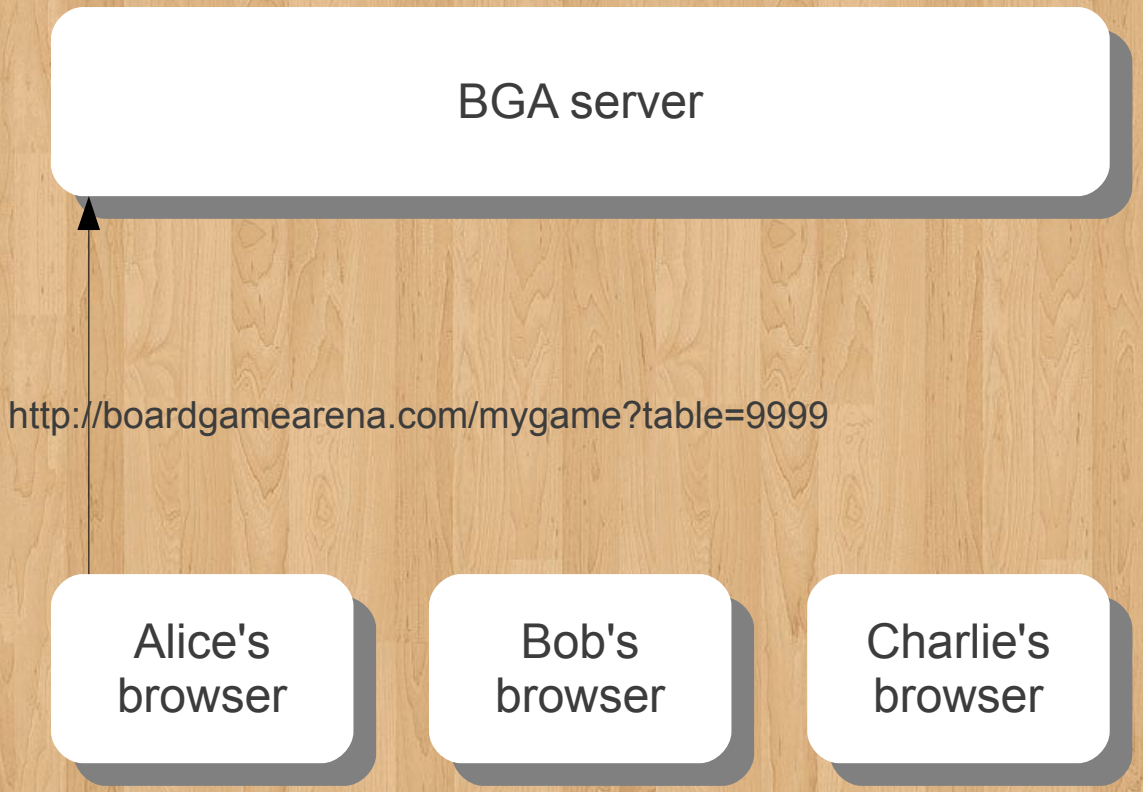
Board Game Arena STUDIO

Loading the game

Alice's browser requests to load the game with the path :

« /mygame?table=9999 »

Where « 9999 » is the identifier of the table.

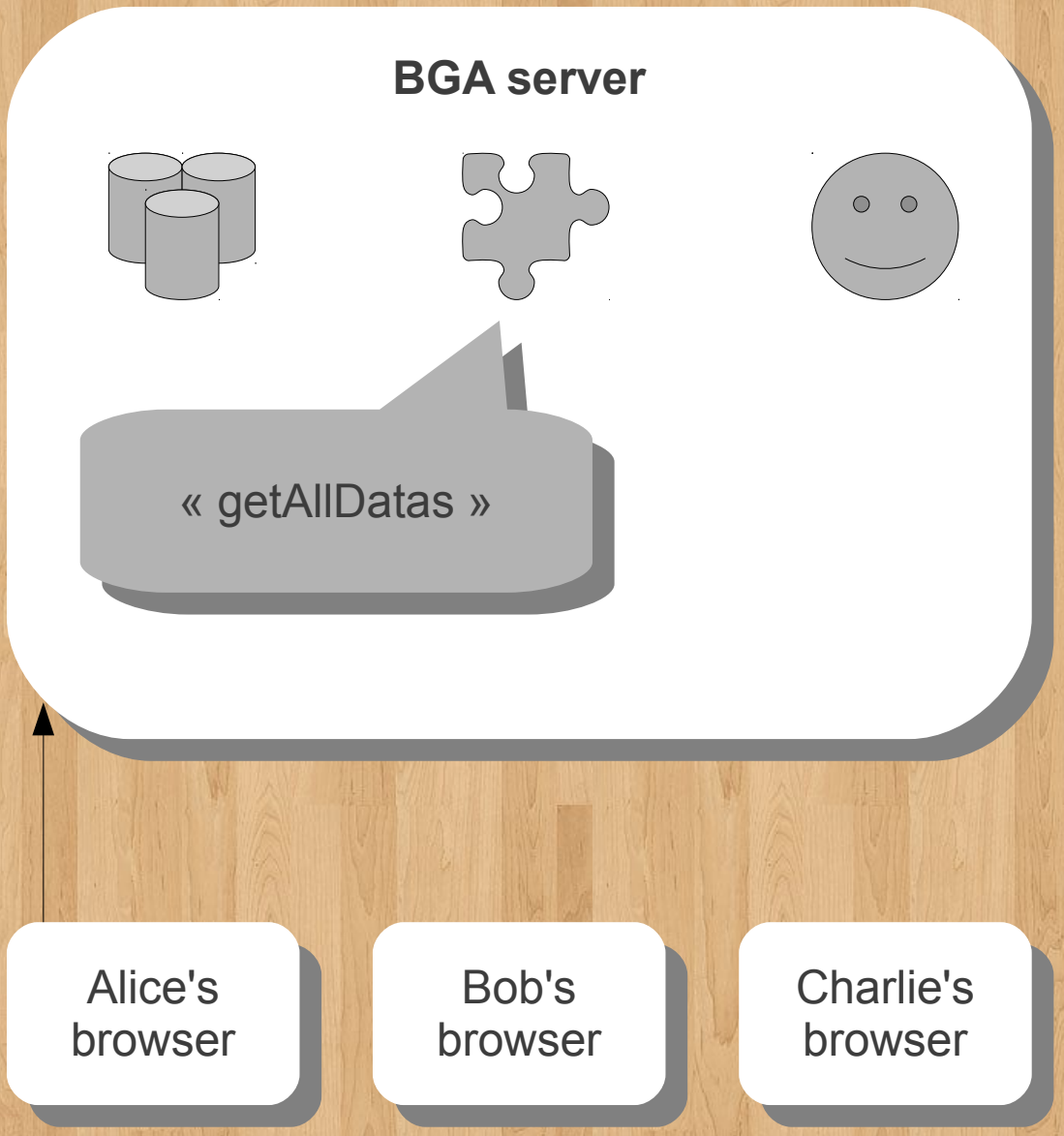


Board Game Arena STUDIO

Loading the game

At first, we have to gather all informations about the current game situation in order Alice's browser can setup game situation on client side.

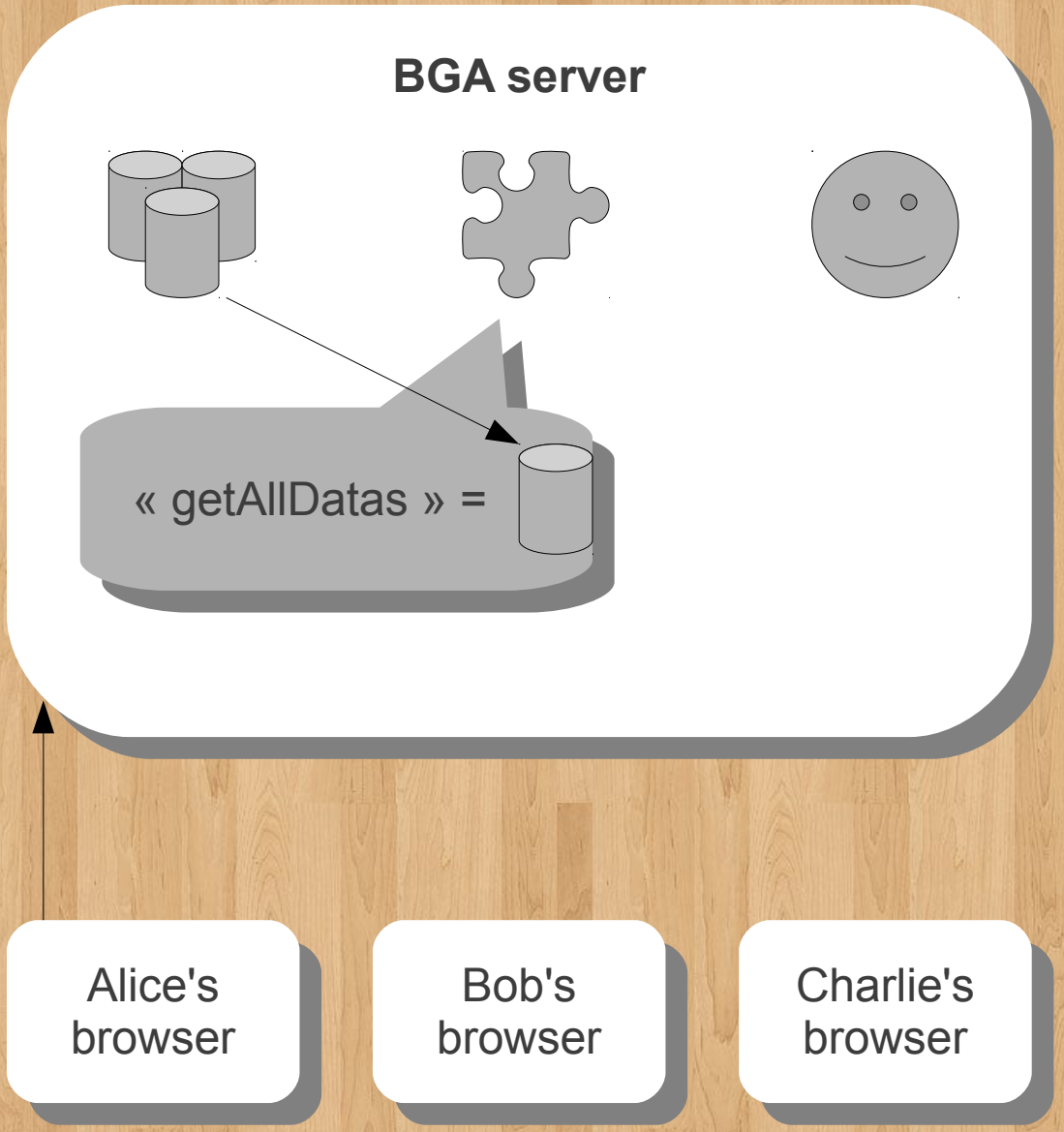
For this, your « getAllDatas » PHP method is called.



Board Game Arena STUDIO

Loading the game

Your getAllDatas method gather all information Alice can see from the game (*and not the cards in the hand of Bob!*), and return this data.



Board Game Arena STUDIO

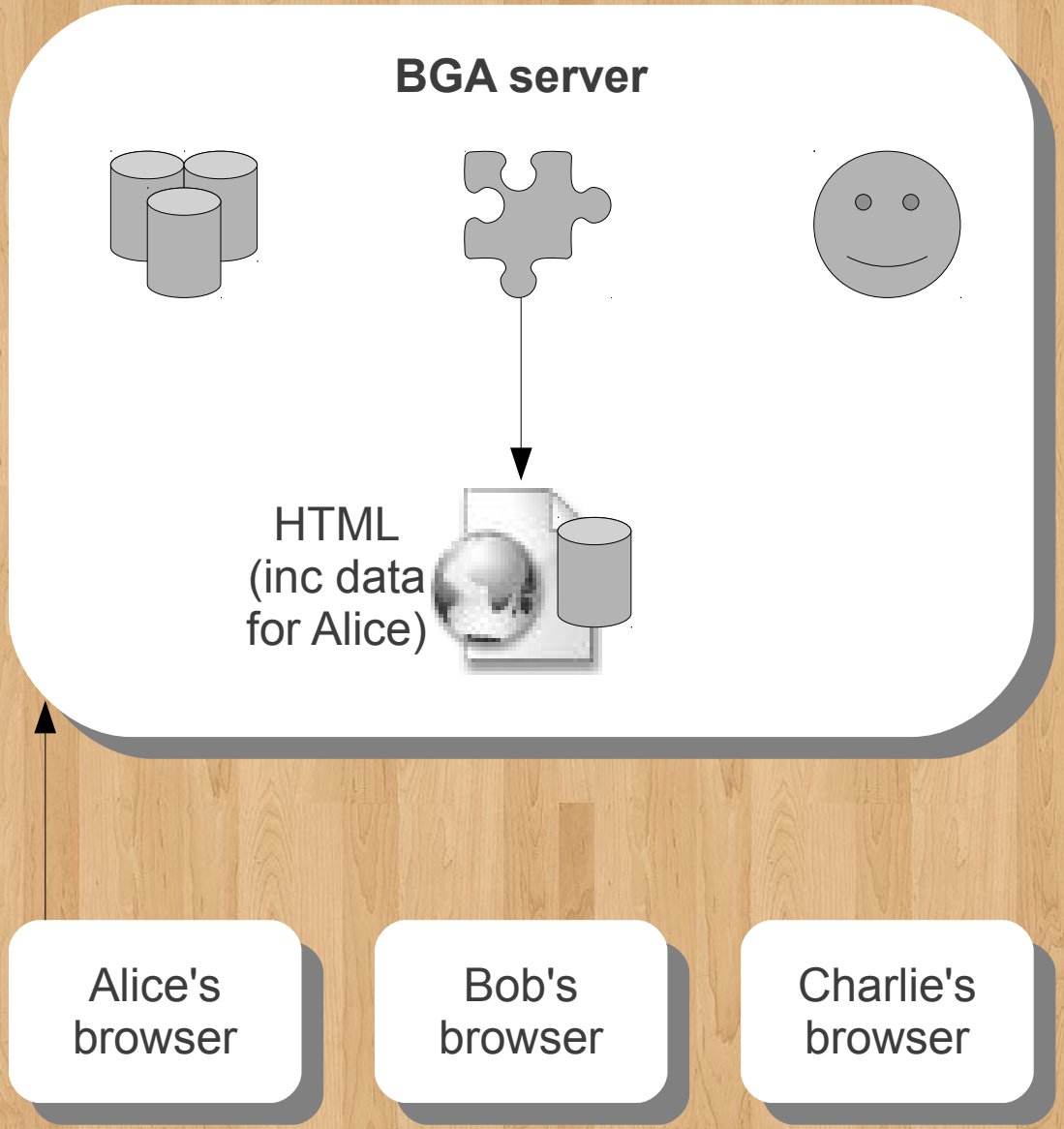
Loading game

Then, we have to generate for Alice a classical webpage where she can play « mygame ». Of course, this webpage is in classic HTML.

This is the **view** of the game.

You can use some logic to create some dynamic HTML page, but most of the time we'll let the client do the biggest part of the game setup and return a simple piece of static HTML here.

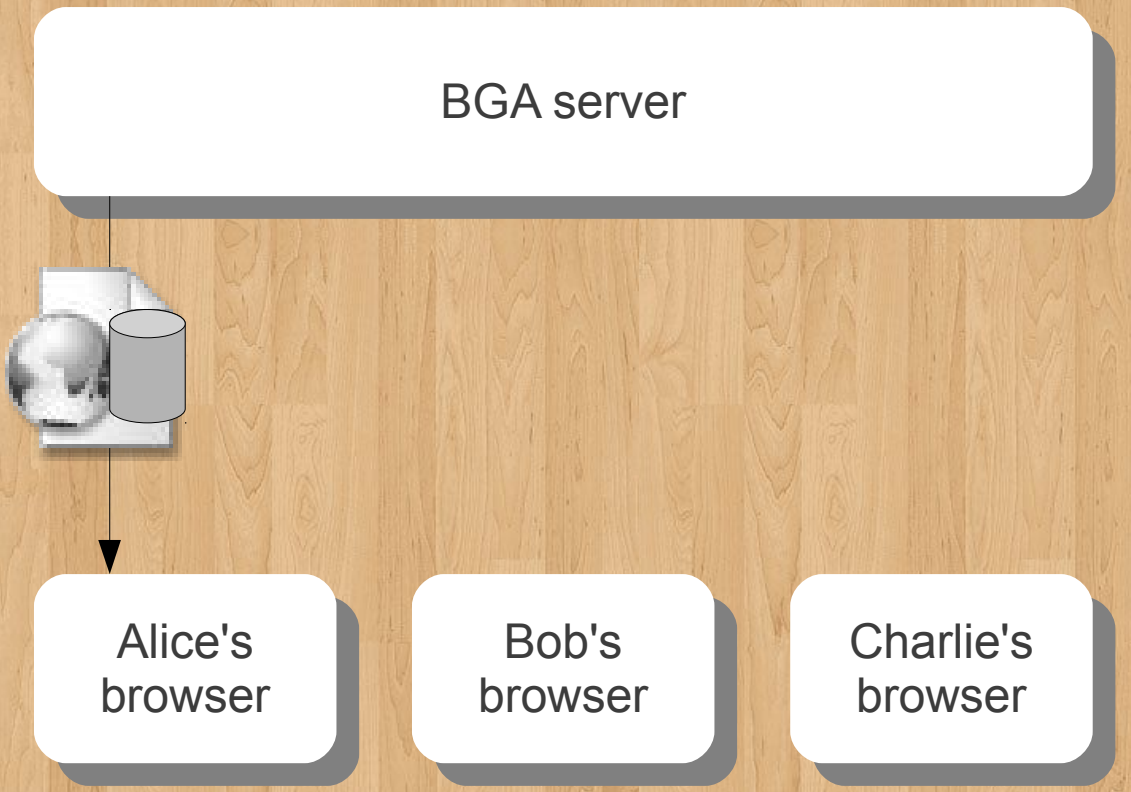
Note that the HTML generated embed automatically the data previously generated (in « json » format).



Board Game Arena STUDIO

Loading the game

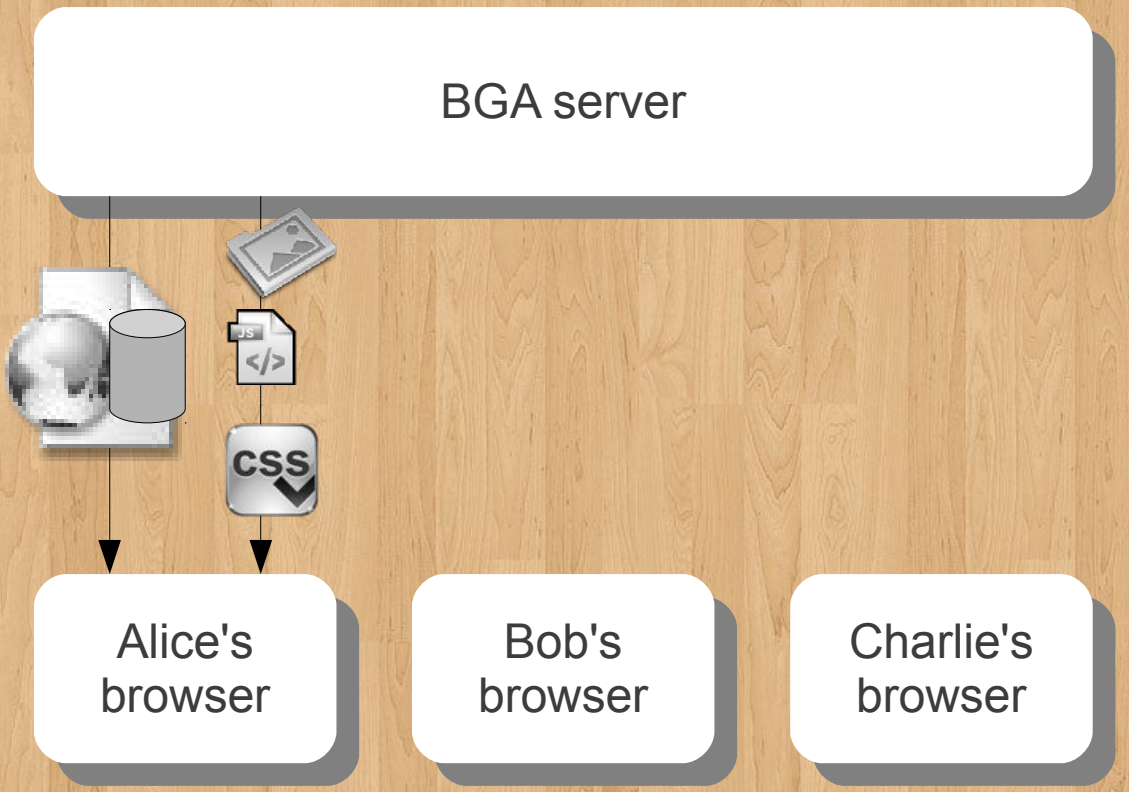
The HTML webpage and all information are returned to Alice...



Board Game Arena STUDIO

Loading the game

Then, like any classic webpages, resources are downloaded : your images, your javascript and your css stylesheet.



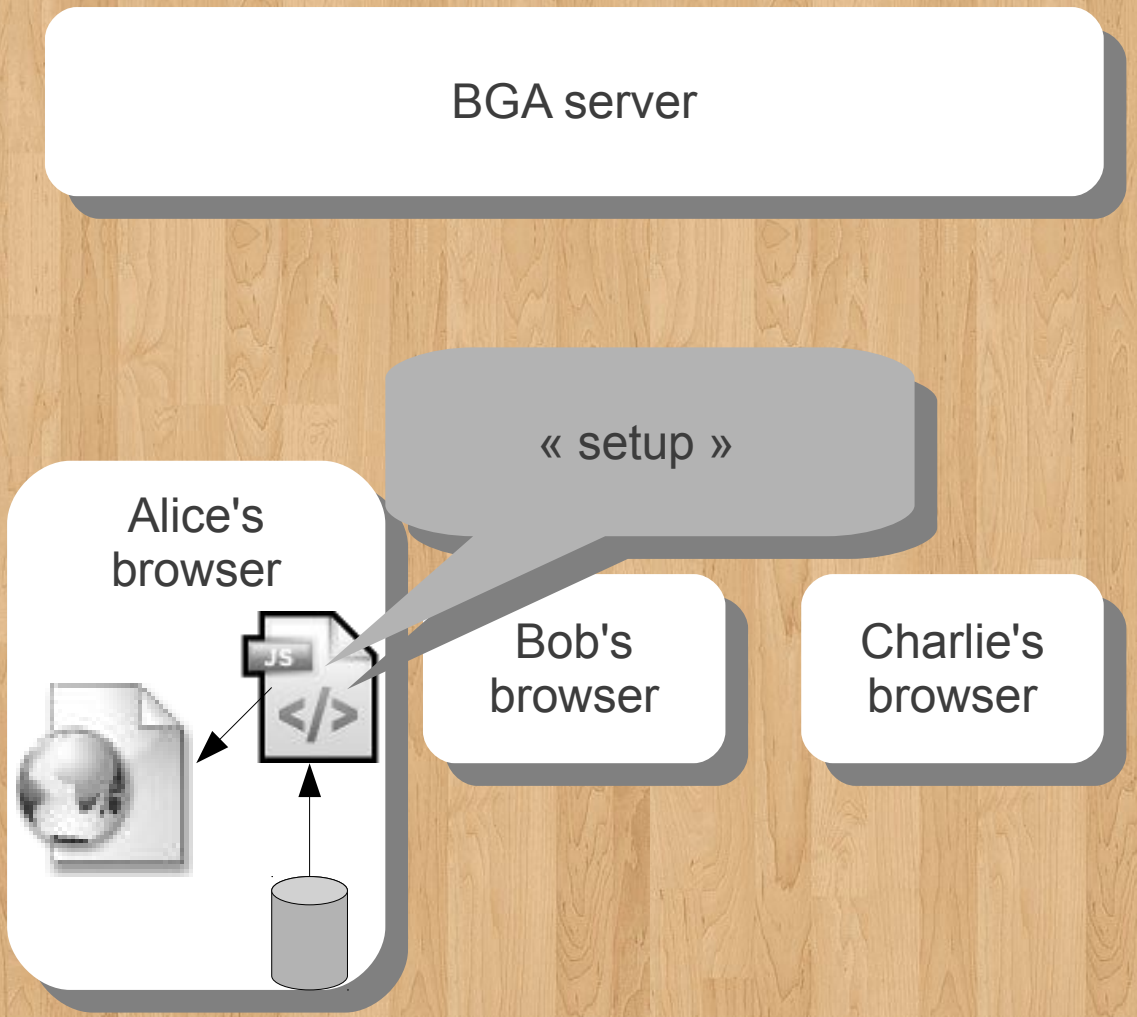
Board Game Arena STUDIO

Loading the game

Finally, your « **setup** » javascript method is called.

In this method, you can use the data returned by the server (« Alice data ») to finalize the game setup for Alice.

Ex : wow, Alice starts the game with 3 money tokens. Let's set her money counter to « 3 ».



Board Game Arena STUDIO

Loading the game

BGA server

This is it !

Alice has now a view of the current game situation, and she's ready to play.

Alice's browser

Bob's browser

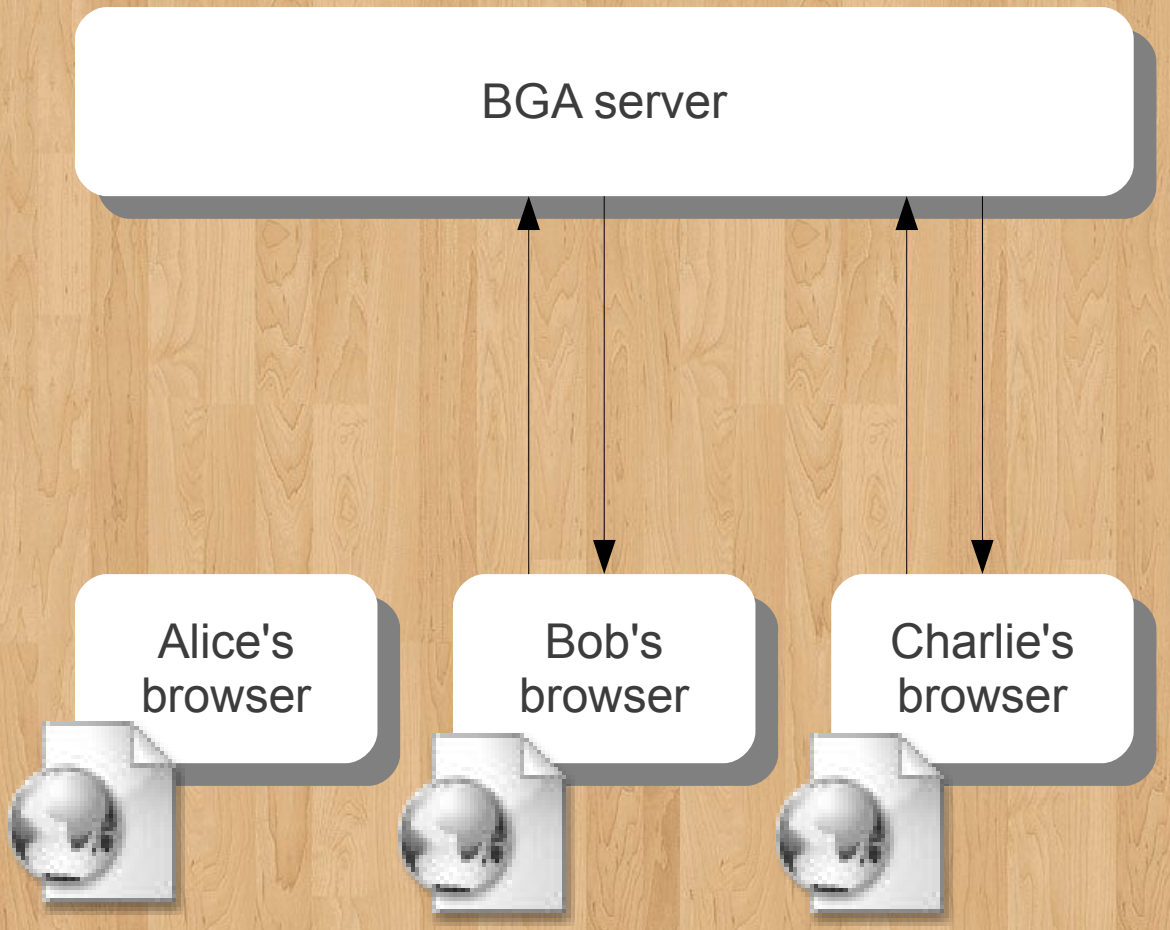
Charlie's browser

Board Game Arena STUDIO

Loading the game

Of course the same thing happened to Bob and Charlie.

Now, everyone is ready to play!



Board Game Arena STUDIO

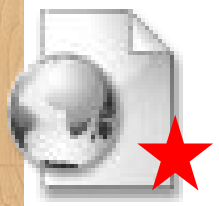
Making a move

It's Alice turn. She is the « **active player** ».

Let say Alice want to play a card, and then clicks on a card.

BGA server

Alice's browser



« clic »

Bob's browser

Charlie's browser

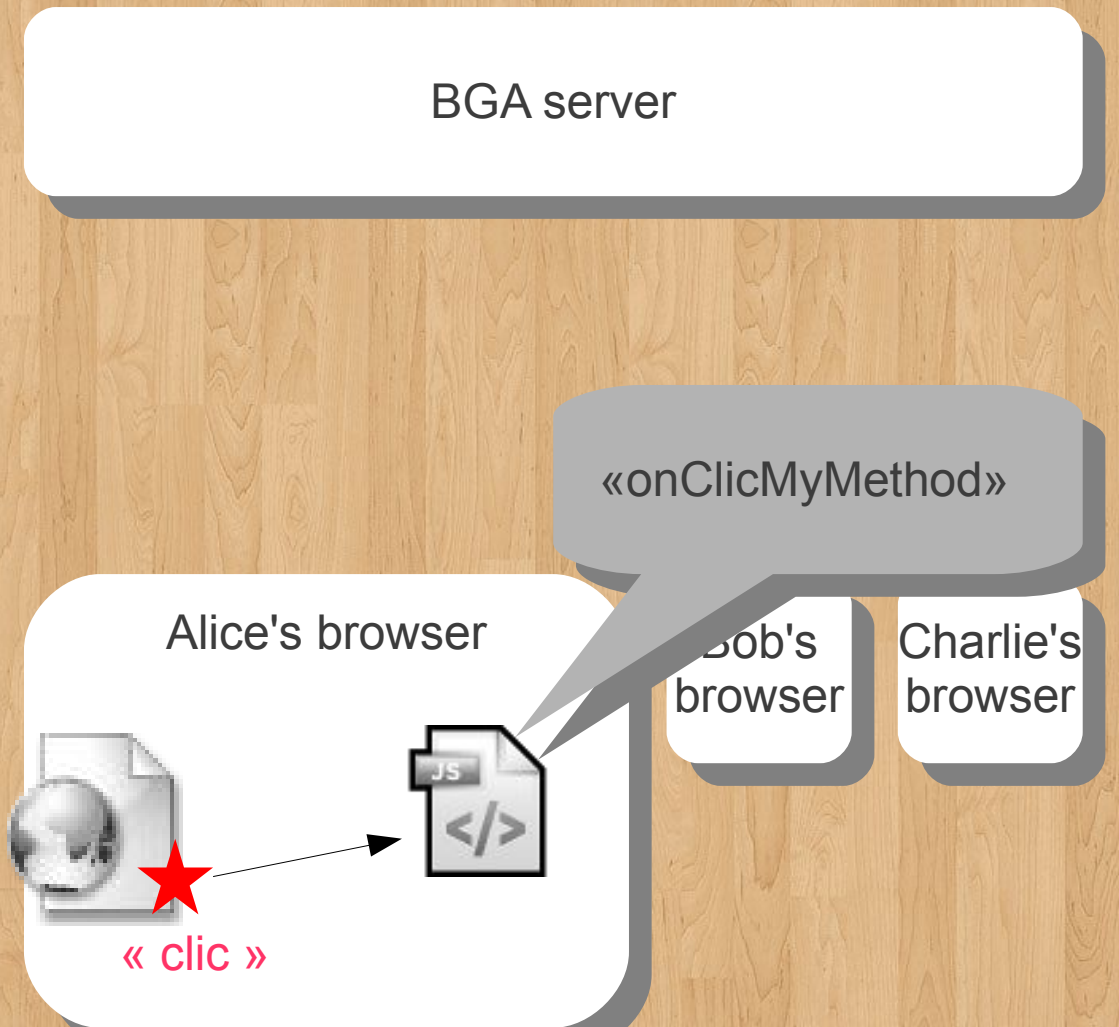
Board Game Arena STUDIO

Making a move

During the initial « setup » of the page, we setup a handler for this « clic » event for this card.

Let say this handler is a javascript method call « OnClicMyMethod ».

OnClicMyMethod is called.

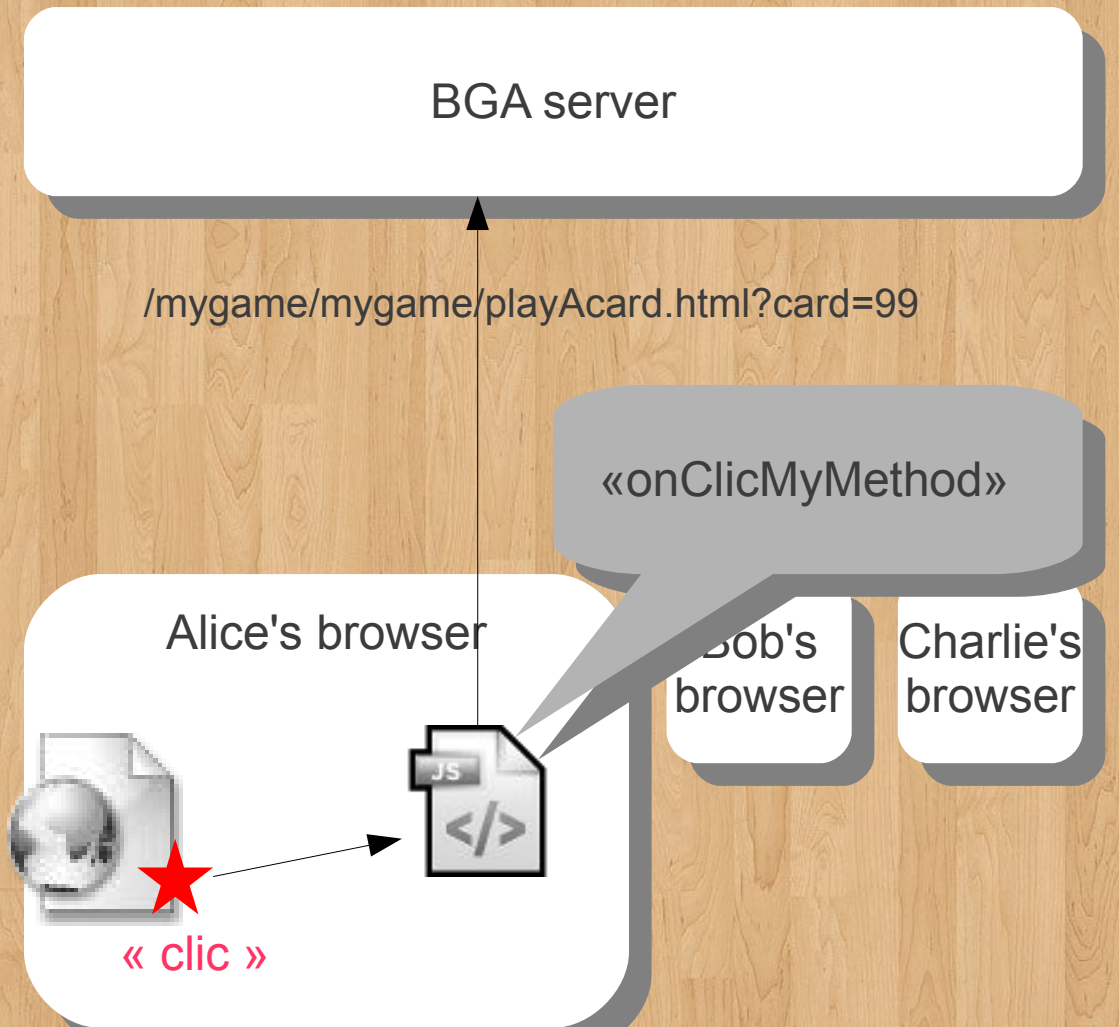


Board Game Arena STUDIO

Making a move

OnClicMyMethod gets the ID of the Alice's card (ex : 99), and send a request to BGA server at this url :

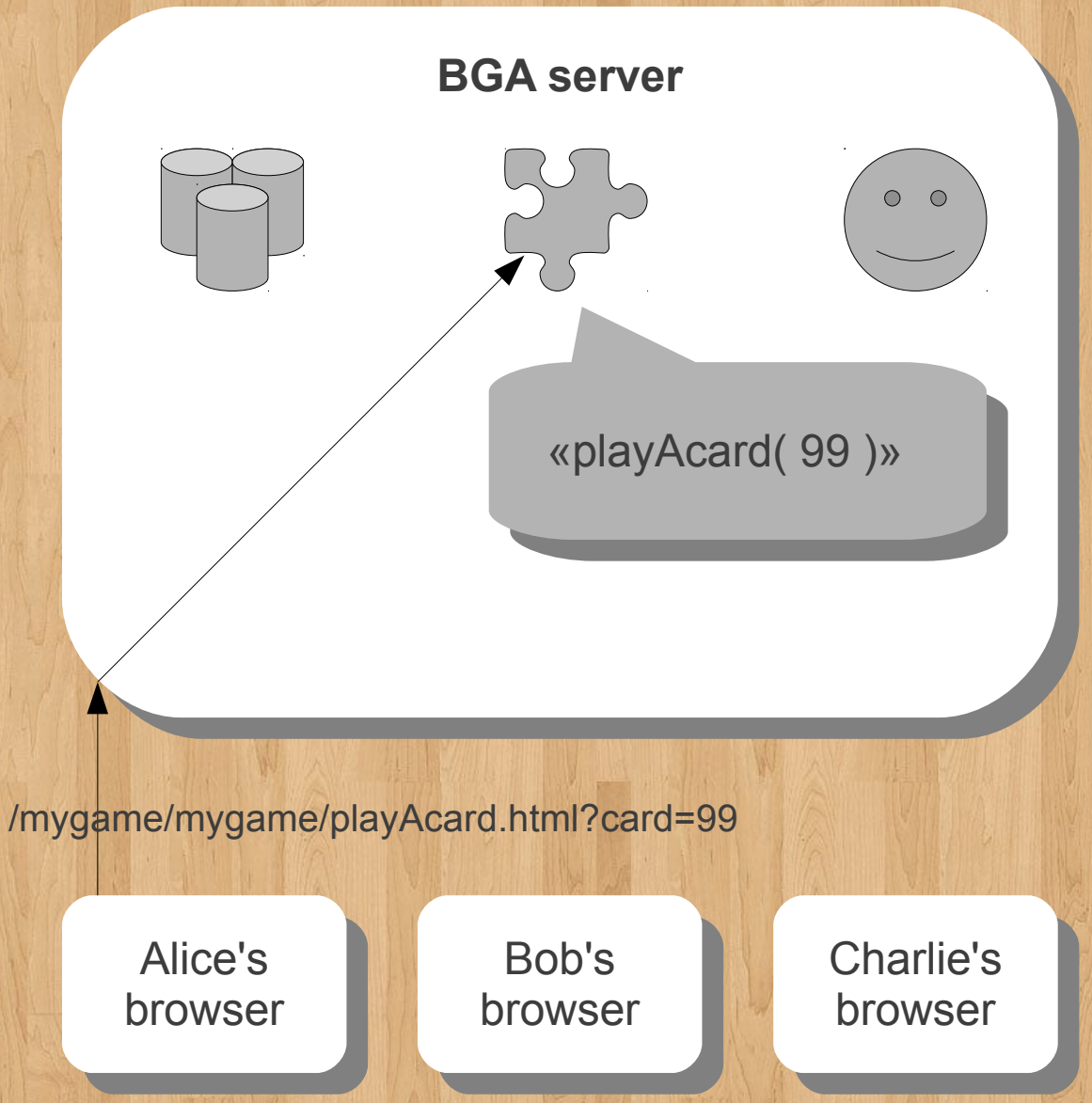
/mygame/mygame/playAcard.html?card=99



Board Game Arena STUDIO

Making a move

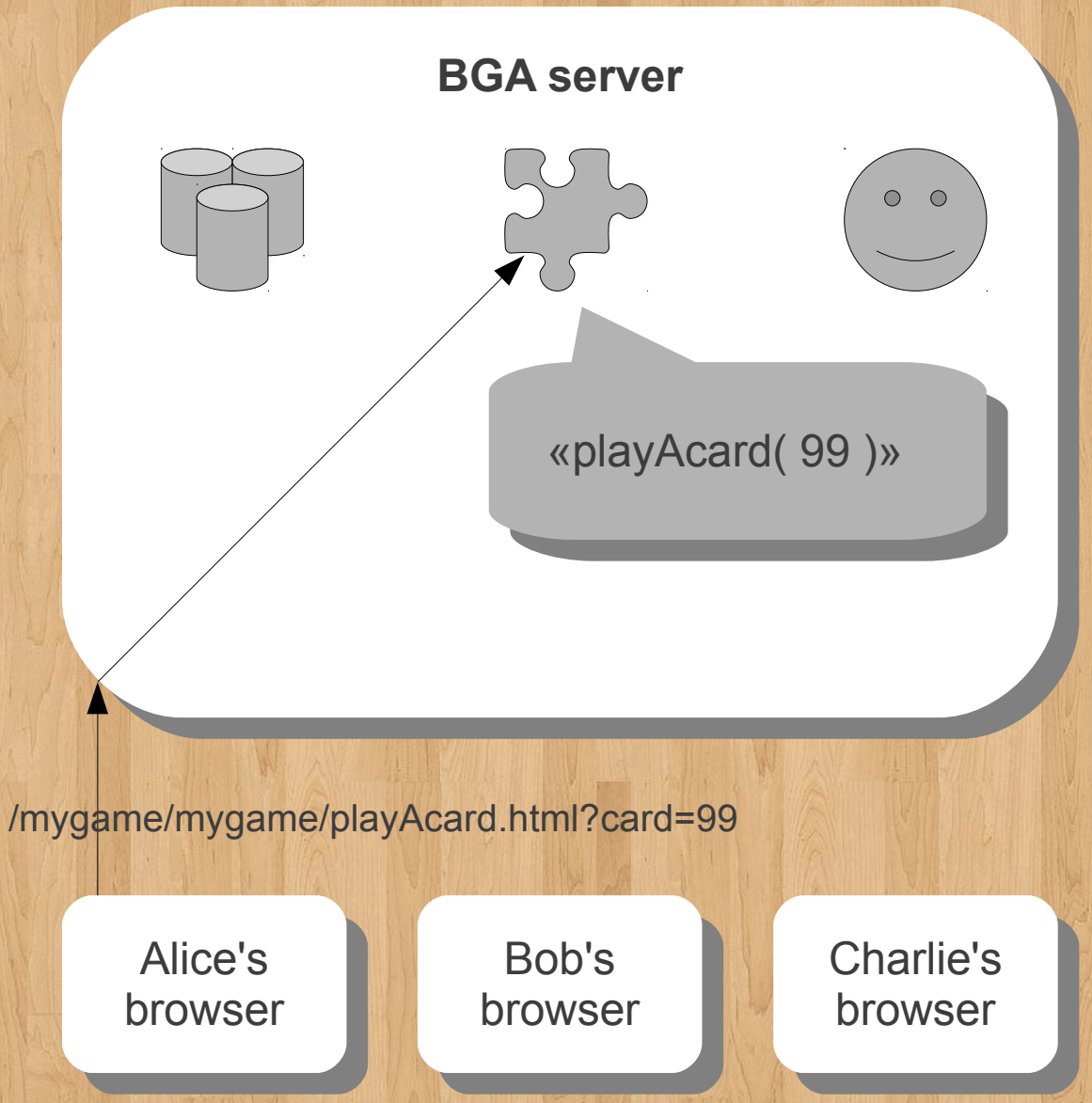
On server side, your corresponding method « playAcard » is called, with the id of the card played in parameter.



Board Game Arena STUDIO

Making a move

Your first work, on this method, is to check if this is a right move :
- Is Alice the active player really ?
- Can she play a card at this moment of the game ?
- Does she really have this card in hand ?
- ...



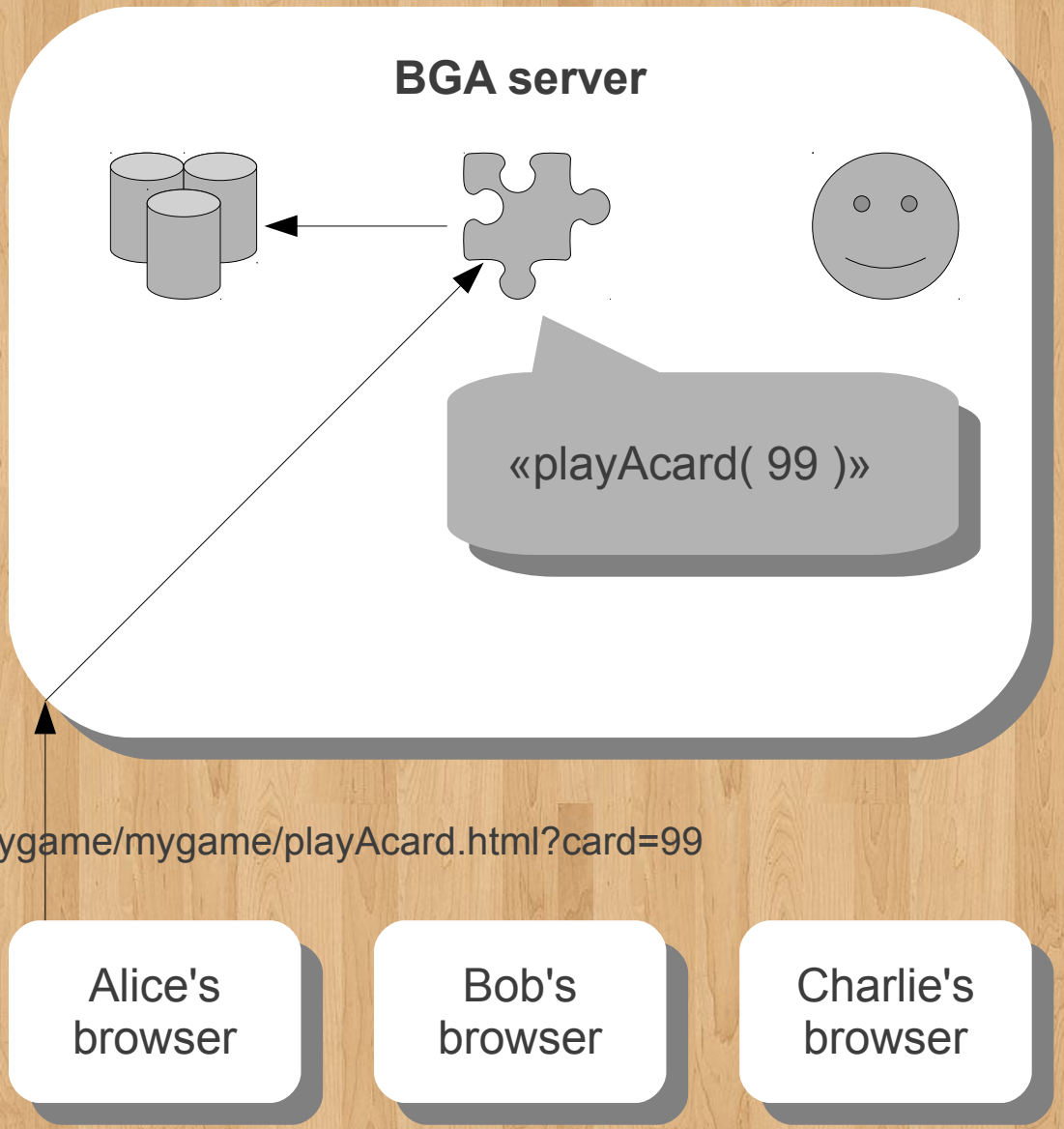
Board Game Arena STUDIO

Making a move

Done !

Now, you just have to apply the rules of the game to the current situation.

Let's say the card played by Alice gives her 2 money tokens. We write in the database that she has 2 more money tokens and that her card is now discarded (and that Bob is the new active player).



Board Game Arena STUDIO

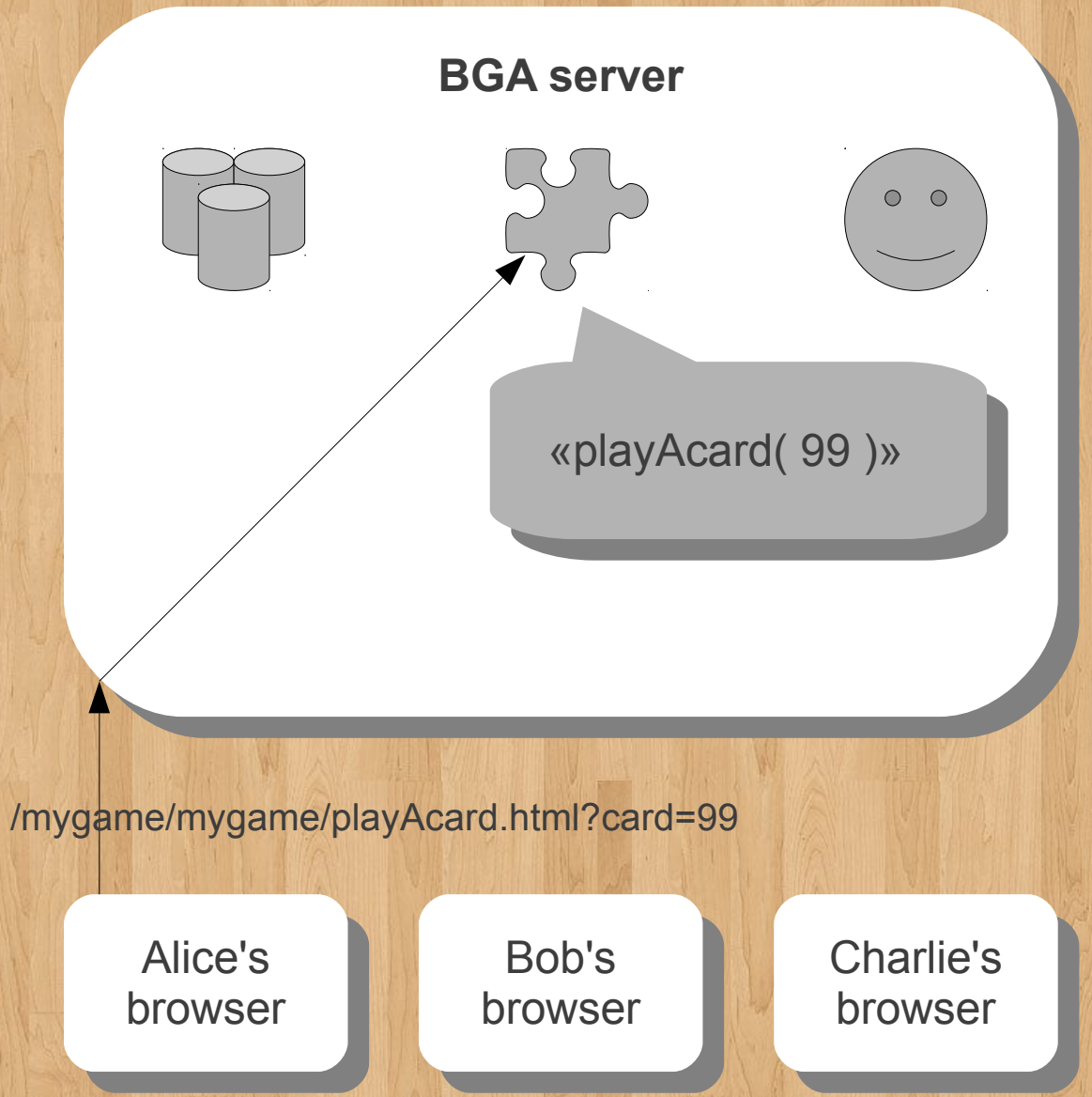
Making a move

Now, we have to **notify** all the players that the situation has evolved.

Note that we can't notify only Alice here. All players must be notified that she has played a card and got 2 money tokens.

The BGA framework proposes a simple method to create and send notifications to players : « notifyAllPlayers ».

Now let's notify all players that Alice got 2 money tokens...



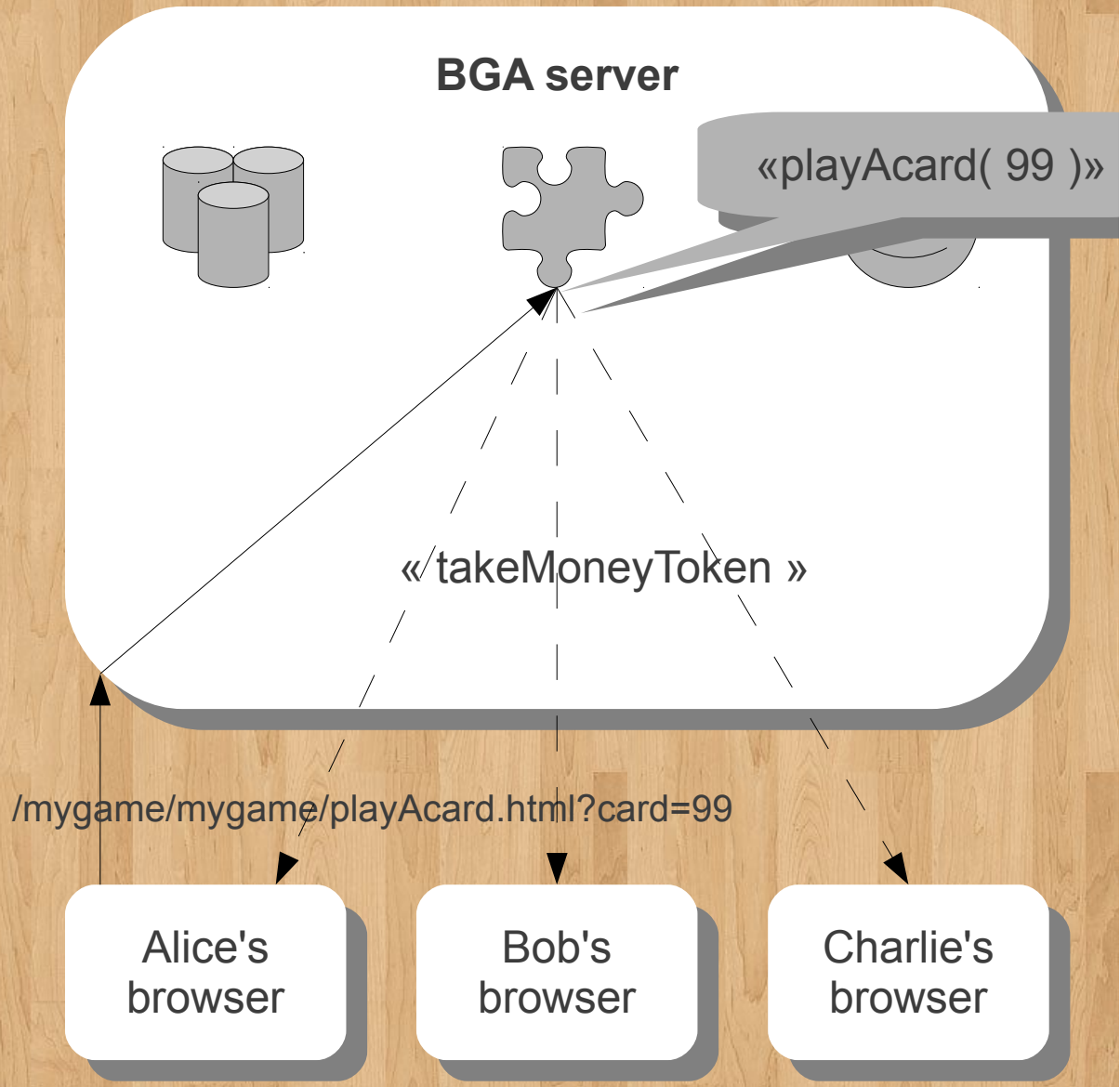
Board Game Arena STUDIO

Making a move

Let's name our notification « takeMoneyToken ».

We associate to the notification a little packet of data saying that the concerned player is Alice and that the number of token is 2.

The notification is sent to all players.

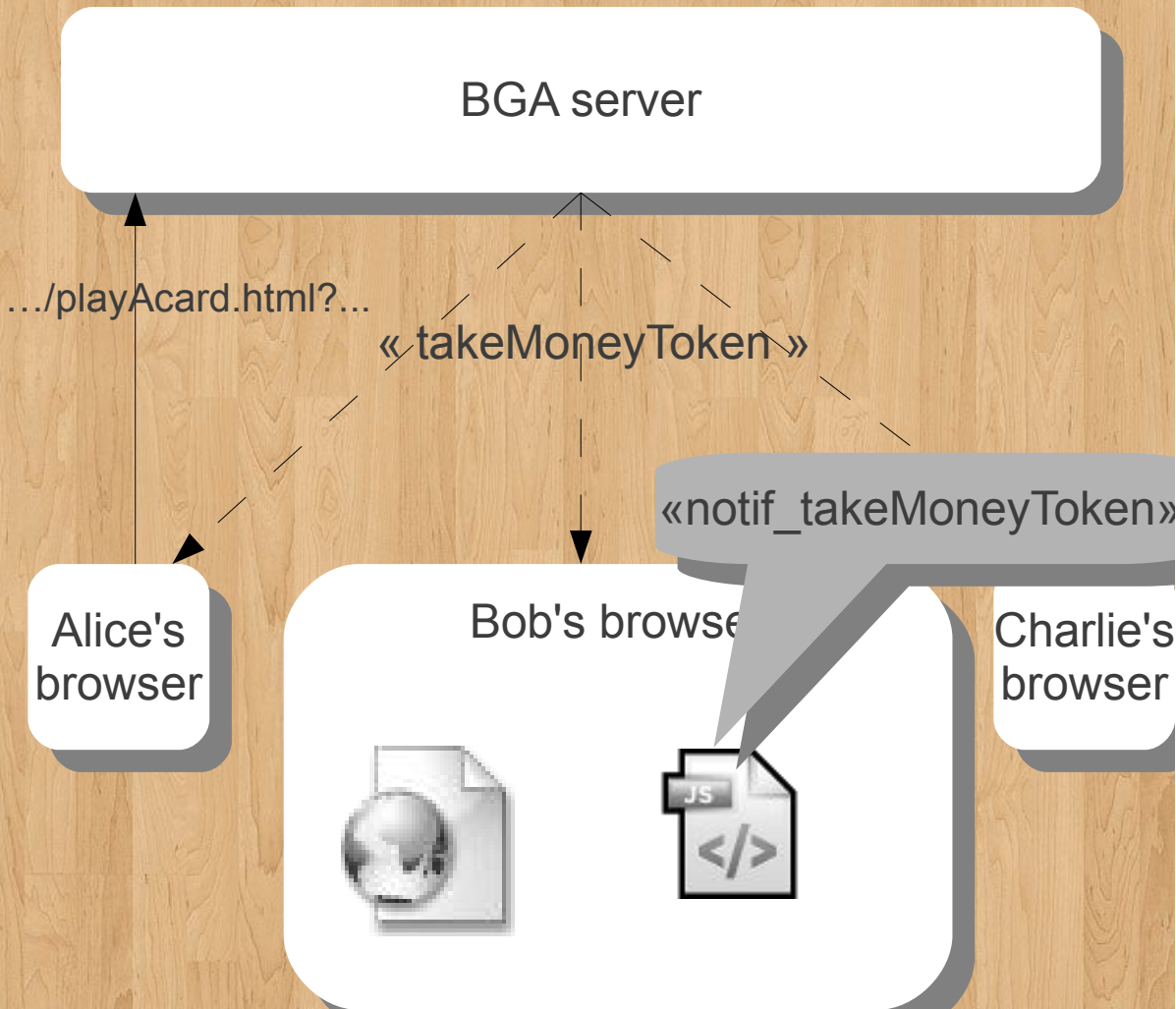


Board Game Arena STUDIO

Making a move

Let's have a look on what happend on Bob's browser.

He receives a « takeMoneyToken » notification. Automatically, your associated javascript method « notif_takeMoneyToken » is called, with arguments saying that it concerns Alice and that the number of money tokens is 2.

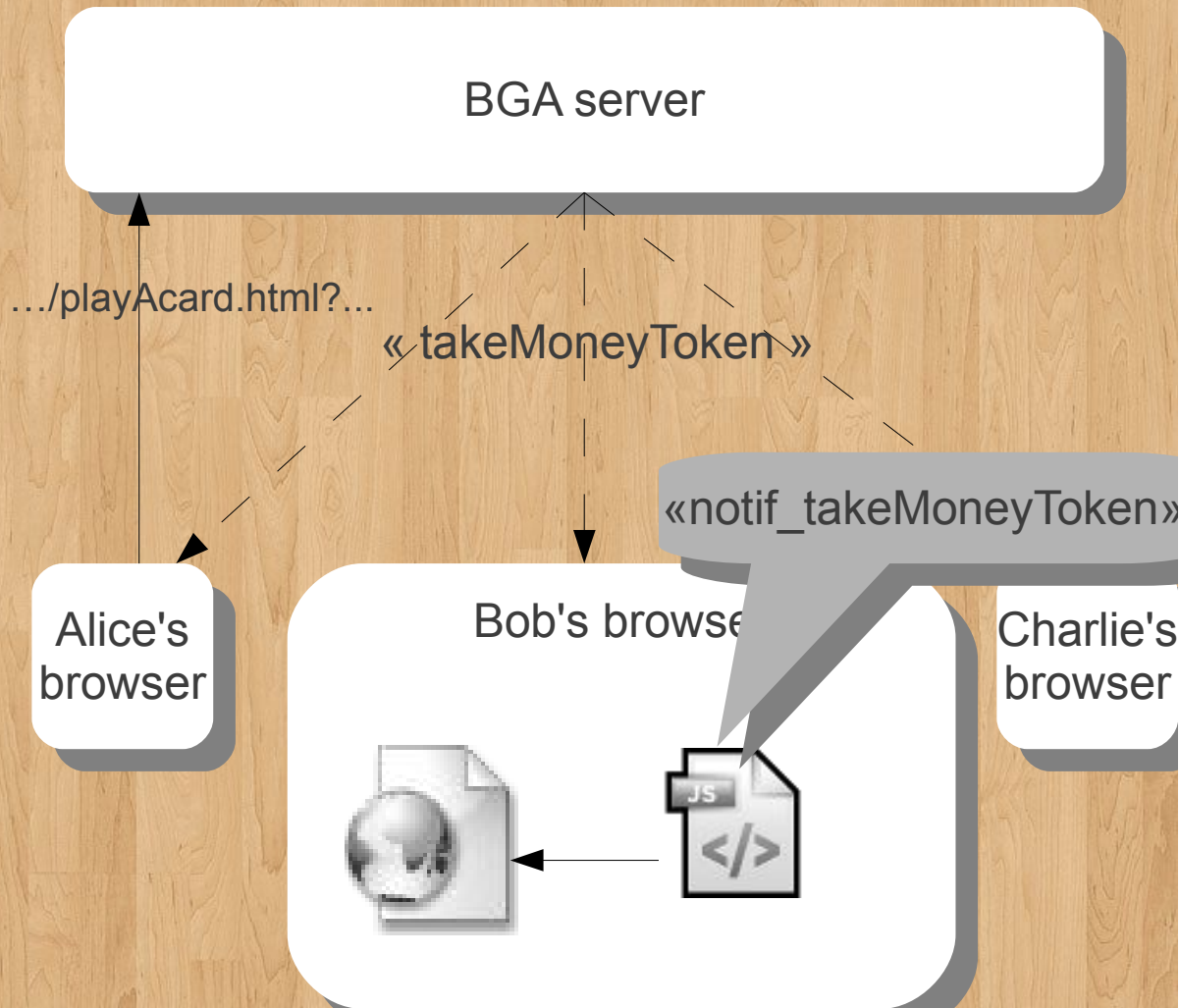


Board Game Arena STUDIO

Making a move

… and finally, your notif_takeMoneyToken just have to increase money tokens number of Alice by 2 on the web page.

Of course the same thing happens on Alice's and Charlie's browser, so everyone knows that Alice is wealthier.

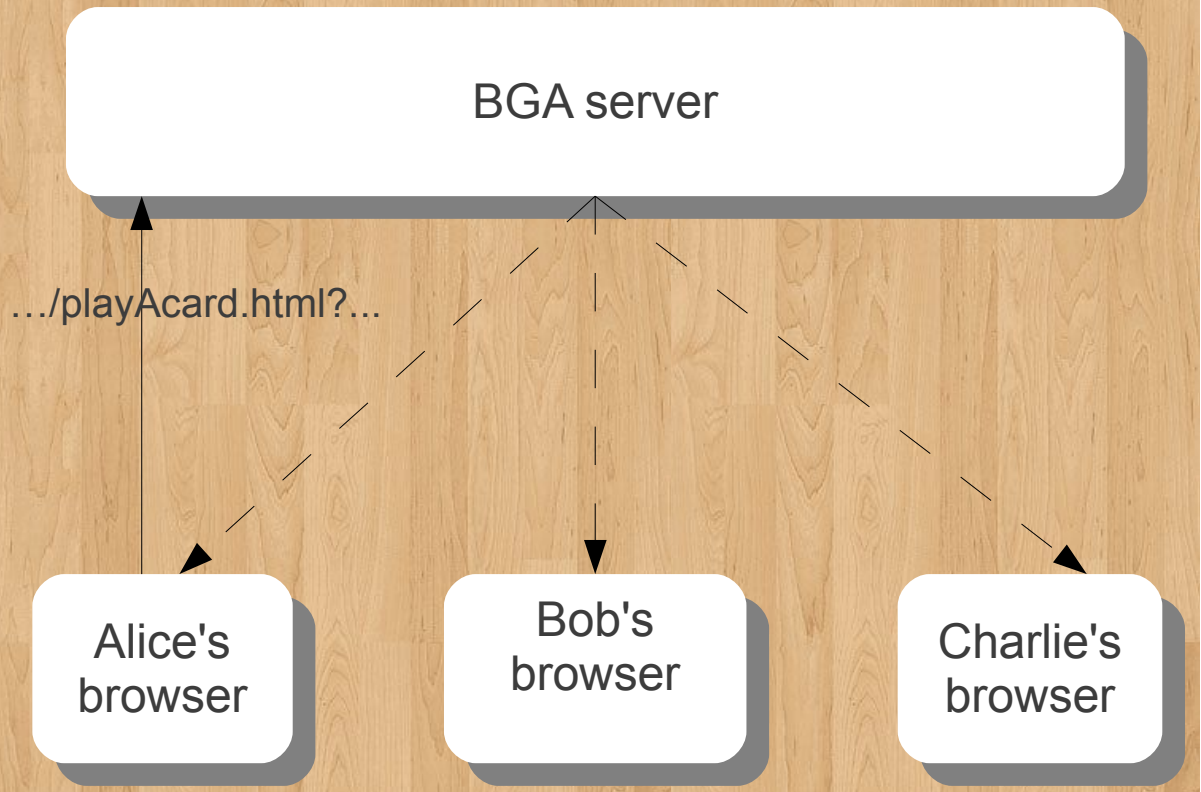


Board Game Arena STUDIO

Making a move

Afterwards, two additional notifications are sent to notify players that Alice's card has been discarded and that it's now Bob's turn.

And this is it : Bob can take some action, and so on...



Summary

• You know what is managed on the server side, and what is manager on the client (browser) side.

• You know which technology is used for each part of the game adaptation.

• You know what happened during basic steps of the game lifecycle : game setup, page load and making a move.

• You understand the concepts of **game database**, **game view**, **game action** and **notification**.