Connect Four

Problem:

People want to play Connect Four with each other through different means, like text or web application.

APIs:

1. Twilio - Allow user to play Connect Four through text.

2. Facebook - Login with Facebook so user doesn't have to create and remember another username and password.

Events:

Clients send events to the game server signaling players' moves. The game server sends events to the clients telling them the state of the game.

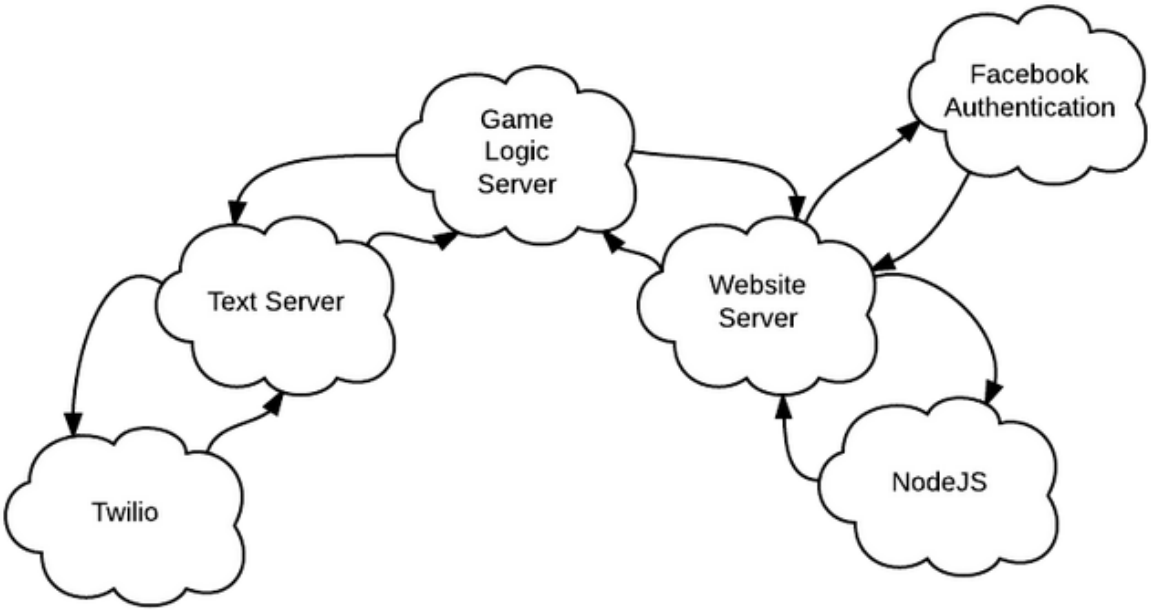
Client sends the following events to the game server:

- _makeMove
    - esl
    - number
- _startGame
    - esl
    - human (boolean)
- _boardRequest
    - esl
    - HTTP Response:
        - player = 0|1 (this correlates to whichever esl you passed in)
        - board = json array[columns][rows]
        - Example Parsing board with php
            - $result = json_decode($result);

The game server sends the following events to the clients, as identified by their esl.

- _moveMade
    - board [5][5] 0,1,-1
- _gameStarted
- _gameEnded
    - victory: boolean

Architecture:



Methods Used:

- Game Logic Server
    - PHP, Yii Framework
- Web Game Server
    - PHP, Yii Framework
- Text Game Server
    - Ruby, Sinatra

Analysis of Event-Driven Architecture:

The event-driven architecture was a simple, elegant way to implement a turn-based game like Connect Four. The clients did not have to match players together or keep track of the state of the game, making it very easy to write clients. The game server did not have to gather user input or present the data to the user. So, essentially, the game server abstracted the game duties out from the clients, so that players from the web app and through text could still play with each other. In order to play, clients only needed to know the events to generate and consume, as detailed above. The programmer writing the client did not need to know how the game server worked.