Technical Test

The test is divided in two parts, **each part of the test should use it's own git repository**. You should just send us links to both repos in the reply.

1. Code Quality

We want to evaluate good coding practices in general, nothing ruby/javascript specific.
Check the provided code. It's a ruby/node.js 4 command line script. Your objective is to improve the code quality.
Rewrite the code following code to make it better **(choose only one file, they are the same program)**: https://gist.github.com/plataforma-co/e2caad8722e5767d996b9721281bffac.
You are given complete freedom on how to accomplish this, the only limit being you should not change the expected functionality (it should still log in users using in-memory user/password combinations). You may even change the signatures, rename methods, create more classes, etc.
You'll need to provide us with a github (bitbucket or gitlab are also fine) repository. Each commit should describe what you are improving and why.

2. Web Development

We want to see how do you handle a small web development project, you can choose a programming language and library/frameworks (except for evaluating the poker hands, that you must implement it yourself) if needed, but **the result must be a working website.**
We'd like you to code a small web app that takes two hands of cards and evaluates which one wins using poker rules.
The app must satisfy the following requirements:

1. Cards should come from our dealer service. Check the documentation below.
2. Once you get the two hands you need to display them to the client. You will also display the result (which of the two hands won), and the name of the combination.
3. You should provide two actions, either to deal the next hands, up to when there aren't any more cards on the deck, or to shuffle the deck and start over.

Below you have the documentation for the dealer service and the rules of how to evaluate which hand wins. You need to provide us with a github (bitbucket or gitlab are also fine) repository **with clear instructions on how to set up an environment to test it**. If you want to include any kind of documentation or notes, please add them in a *doc/* directory.

Feel free to take any decisions on things not included here (and explain them in the *doc/* directory if you think it's necessary).

Dealer service

You can reach the dealer service on this url: https://services.comparaonline.com/dealer

The service can't be changed, so you might need to find workarounds for any issues you might face.

The service is designed to fail often (throw a 500 HTTP error), so you need to account for that, the user experience should never be affected.

Here you can find example calls to both endpoints:
https://gist.github.com/plataforma-co/57691938b63c3c29418a9bf5dca7b896

POST /deck

Calling this method shuffles a new deck (52 cards). Decks expired after 5 minutes without being used, so you should handle that possibility. It returns a 36 character token, plain text.

GET /deck/{TOKEN}/deal/{AMOUNT}

This deals an {AMOUNT} of cards for the specified deck {TOKEN}. The result is a JSON array containing objects with *"number"* and *"suit"* properties. "number" is a string that can be one of: A, 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K. "suit" is also a string and can be either hearts, diamonds, clubs or spades.
If there aren't enough cards to deal the amount requested, it will throw a 405 HTTP error.
If the deck isn't found (doesn't exist, or expired) it will throw a 404 HTTP error.

Poker rules

You must implement the logic for ranking hands and checking winners yourself, don't use third party poker libraries.
Each hand consists in a set of **five** cards.
One hand wins if its hand ranking is better.
If two hands are ranked equally, then the hand with the highest value cards wins.
If the two hands are both ranked equally and their highest cards are the same, then the second highests cards are compared, and so on.
If they are still equal, it's a tie.

Hands Ranking:

1. High Card: Highest value card. Order is 2, 3, 4, 5, 6, 7, 8, 9, Ten, Jack, Queen, King, Ace.
2. One Pair: Two cards of the same value.
3. Two Pairs: Two different pairs.
4. Three of a Kind: Three cards of the same value.
5. Straight: All cards are consecutive values.
6. Flush: All cards of the same suit.
7. Full House: Three of a kind and a pair.
8. Four of a Kind: Four cards of the same value.
9. Straight Flush: All cards are consecutive values of same suit.
10. Royal Flush: Ten, Jack, Queen, King, Ace of same suit.