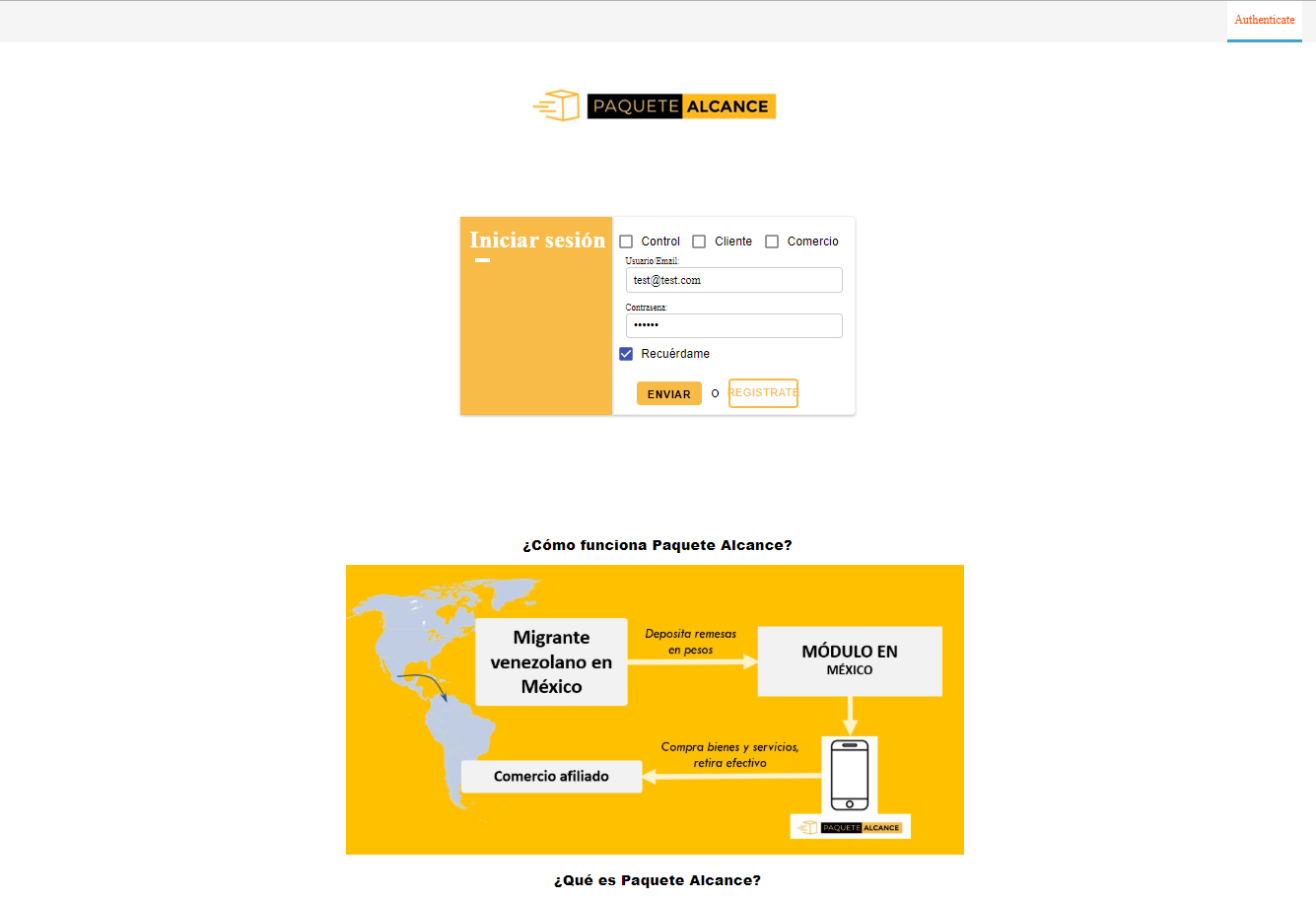
PAQUETE ALCANCE

-Technical Platform presentation description-



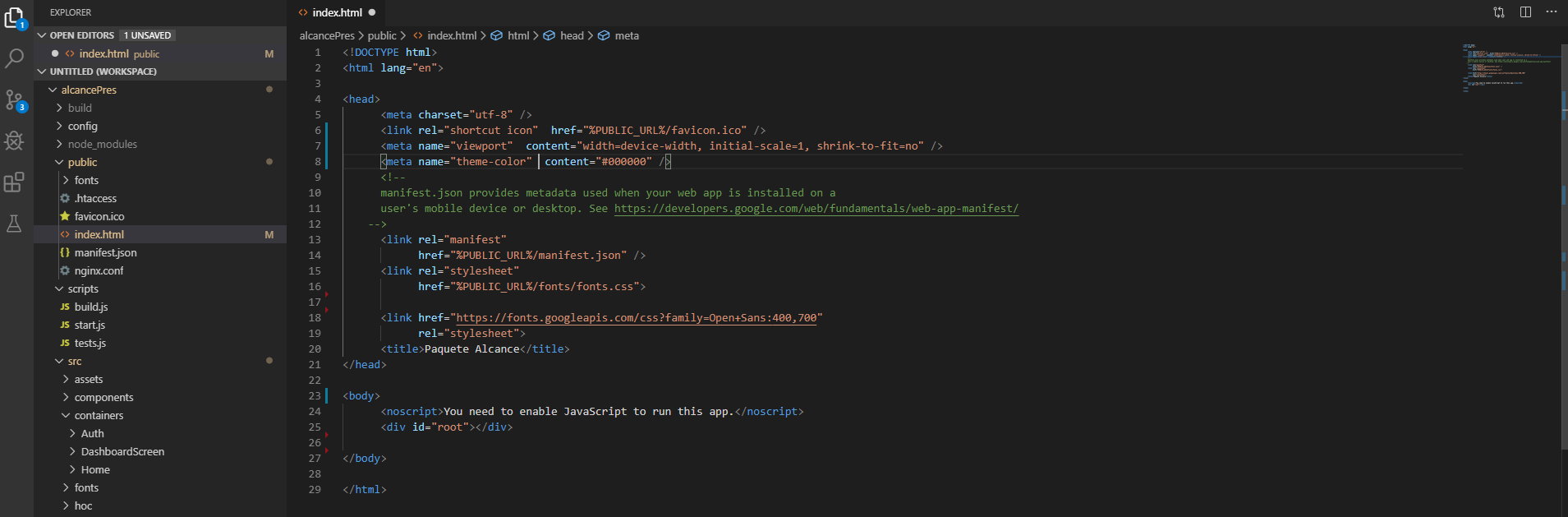
Proposed design as a web application that interacts with the user dynamically, on a frontend multiplatform support architecture oriented for compatibility and accessibility. In extension can be made even a mobile application annexed as user interface connected at main service functionality.

Project platform general architecture:

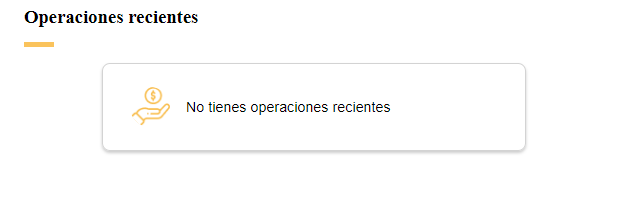
* FrontEnd render architecture
* FrontEnd registeredServiceWorker processing and structure logic
* FrontEnd Information Exchange
* BackEnd data storage and processing

Front-End render architecture

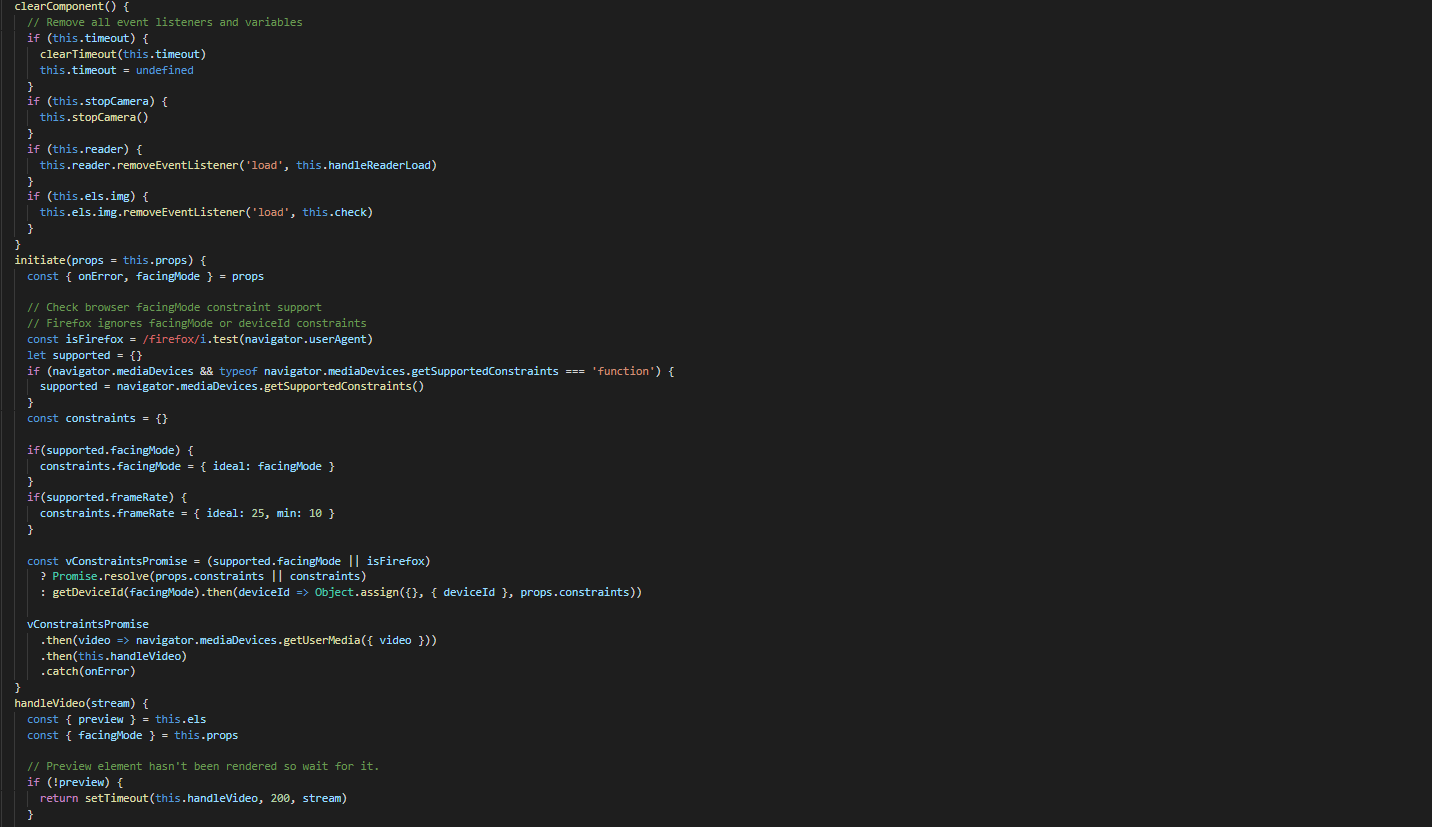
*Multiplatform web interface java script generated render, based on composition of components*.



* General structure architecture designed with containers that permits nesting subcomponents with a time-action dependency that can vary according to display(render) necessity.
* Every item that appears in User Interface (UI) is generated at request by an application back- logic function/functionality, embedded into minimized bundle (webpack usage).
* The user interaction, commands trough button events the required element to be rendered and in real time is generated and displayed.

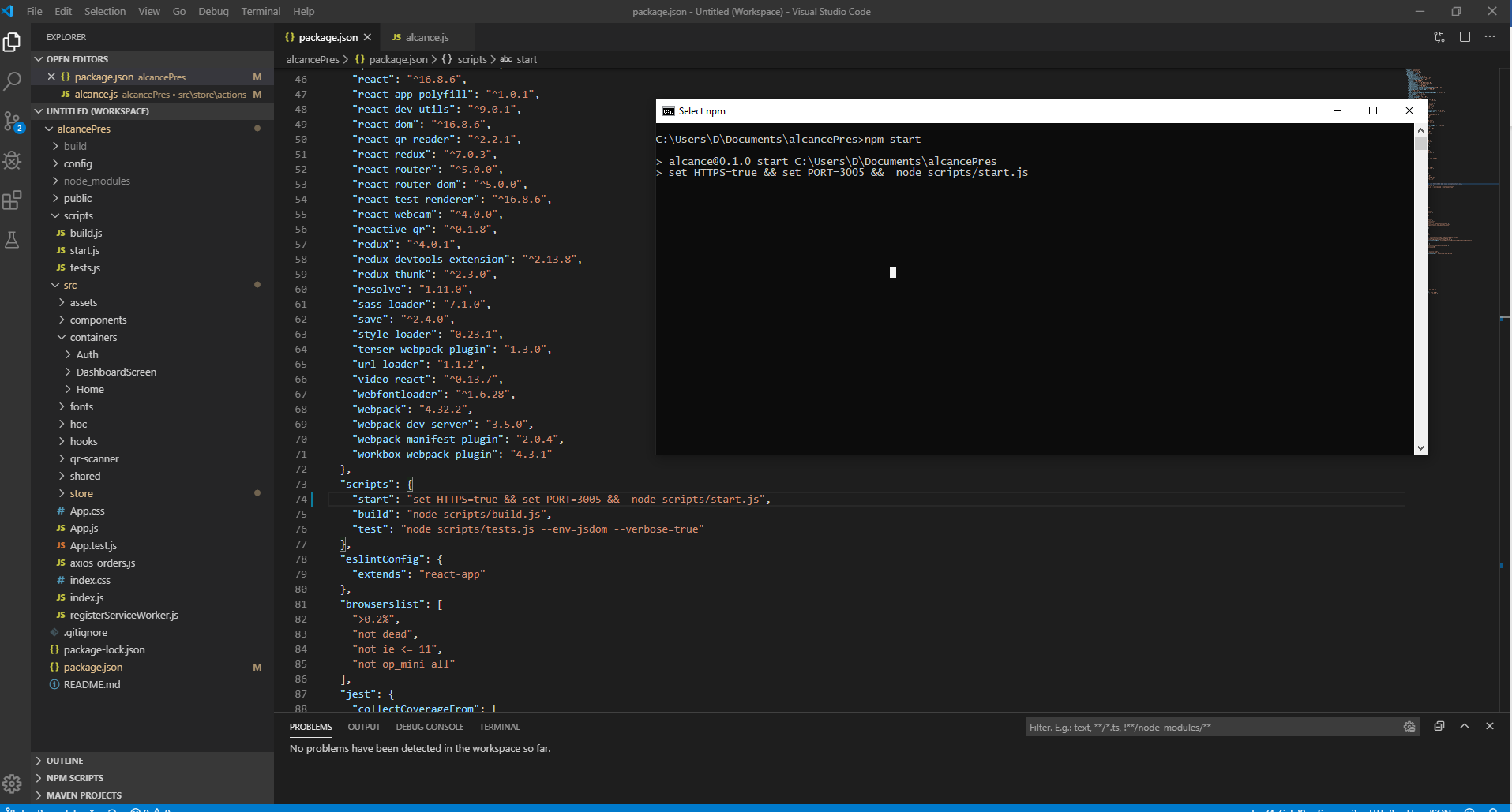


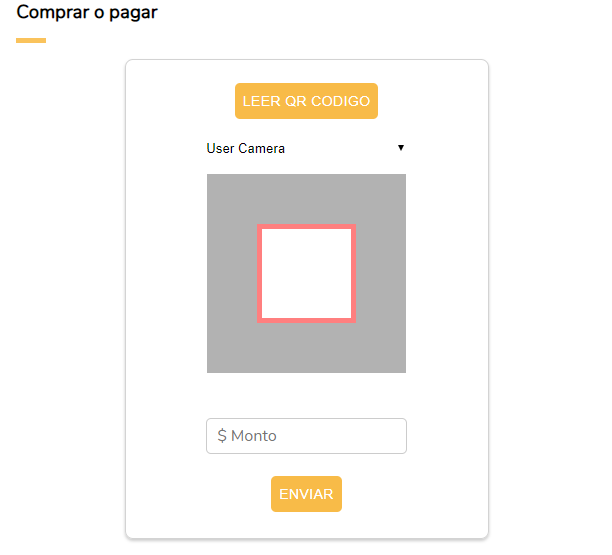
* Every component returns a [description of what needs to be rendered](https://reactjs.org/blog/2015/12/18/react-components-elements-and-instances.html#elements-describe-the-tree), and that description may include both custom components like <ENVIAR> (or <QRCode value={whatToTranscript}/>) and platform-specific components like <div>. It is up to app itself to “unroll” <ENVIAR> at some point in the future and actually apply changes to the UI tree according to the render results of the components recursive.
* When page is changed the loaded elements are automatically destroyed, (not stored and reloaded), such behavior preventing permanent interception of displayed fields by conventional approaches.
* Generating and reading QR codes are self-processing module entities with real time computing functions (implemented registeredServiceWorker) included in minimized bundle:



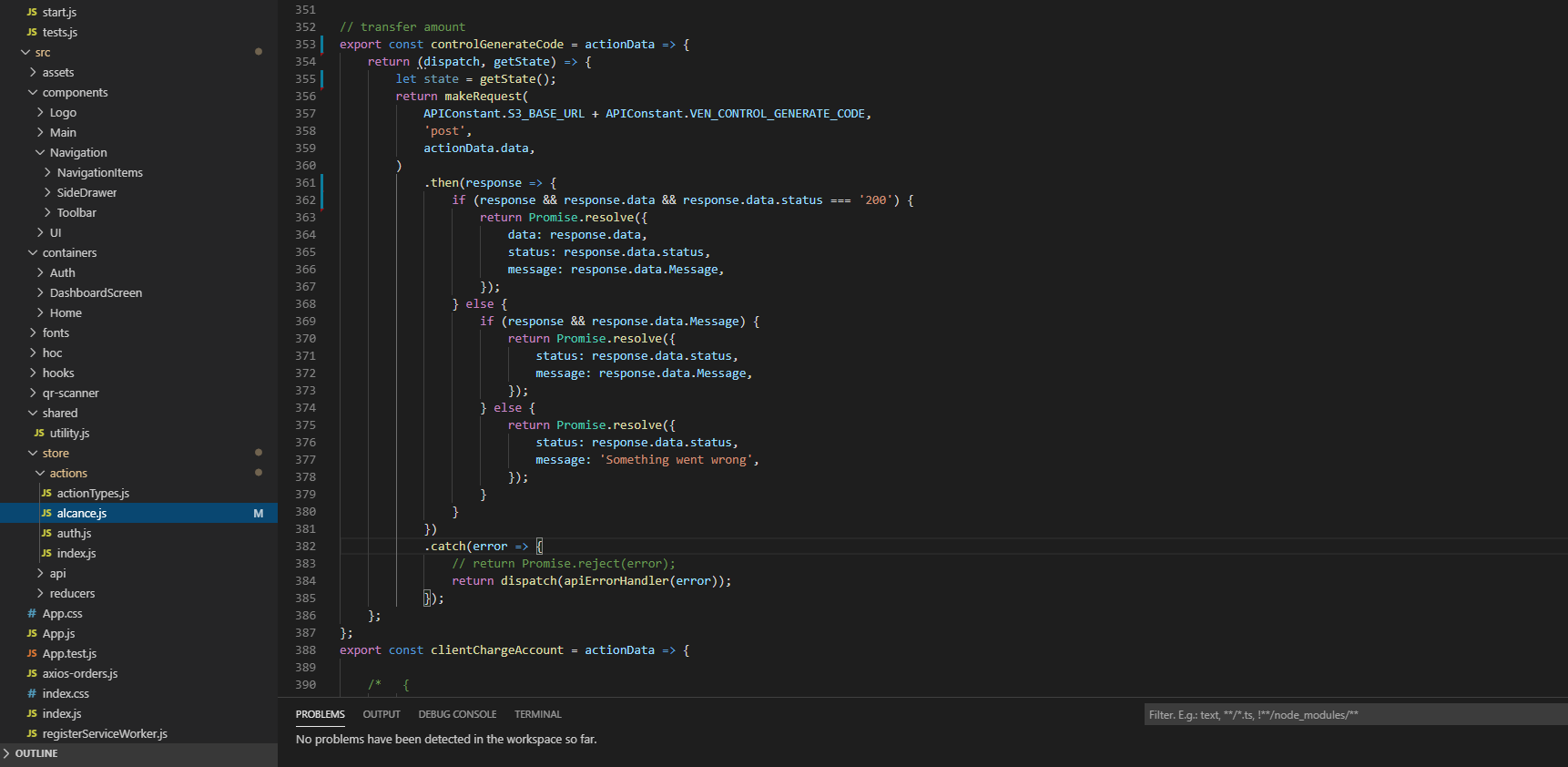


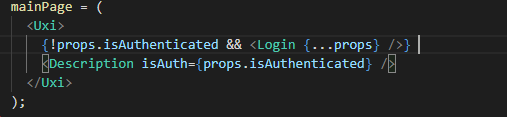
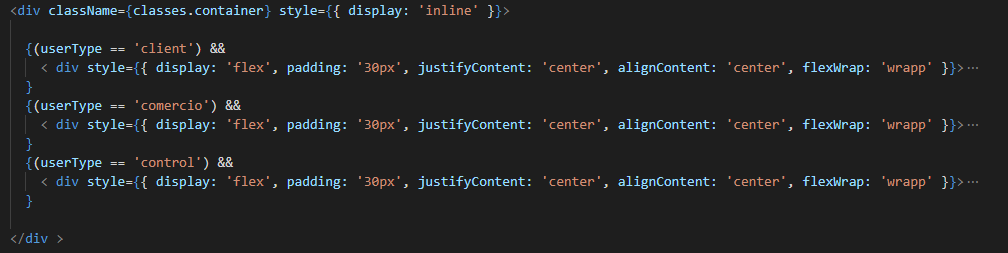
FrontEnd registeredServiceWorker processing and structure logic

* From behavior point of view, working as a self computing application the front-end platform uses modular real-time processing that constraints the environment to use predefined security protocols (https implementation). 
* By abstract if user forces the use in other than recommended environment that does not comply with predefined internal constraint checks, features will not be available, determining technical assistance intervention (user notification about unavailability can be implemented ).

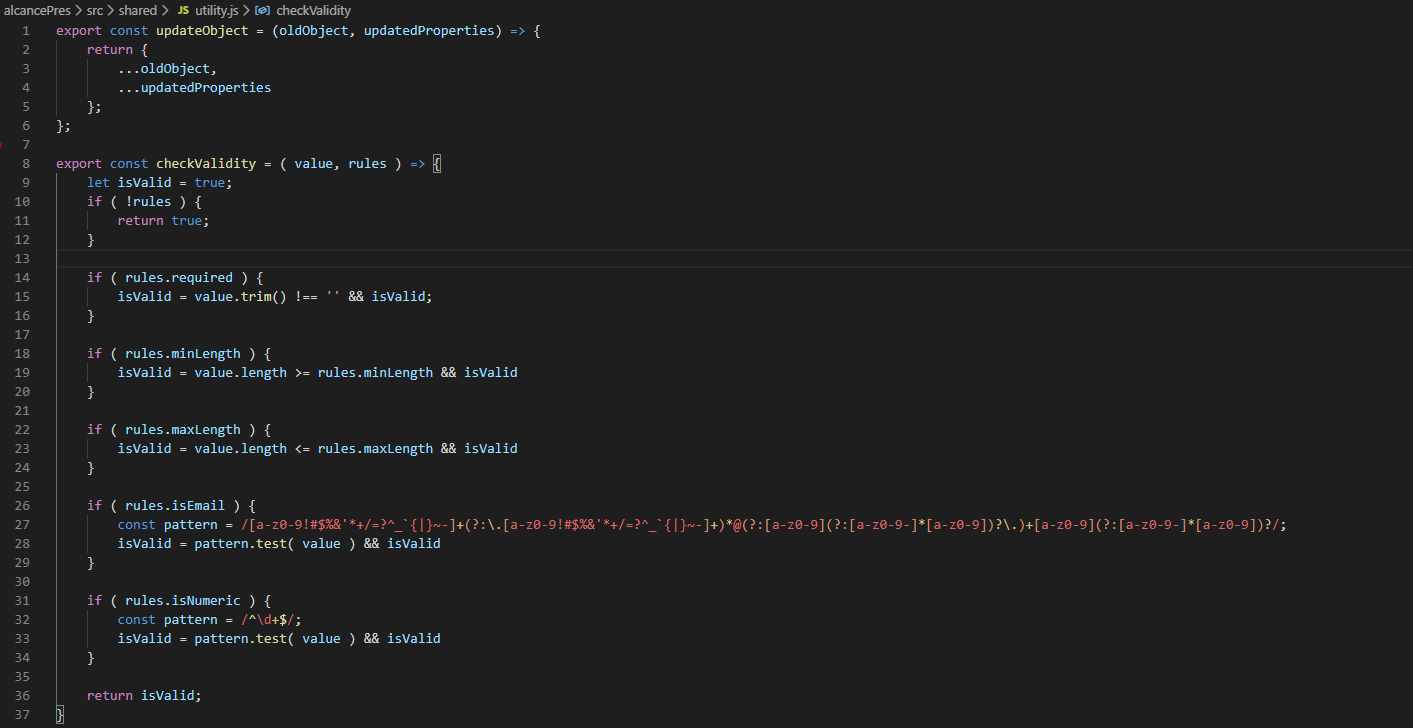


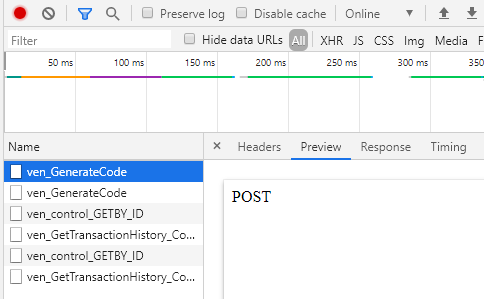
* Procedures of obtaining data have only receive-and display character and not receive and store. Received information from back-end is displayed as it is, with volatile character, being destroyed at refresh or navigation to other interface section.



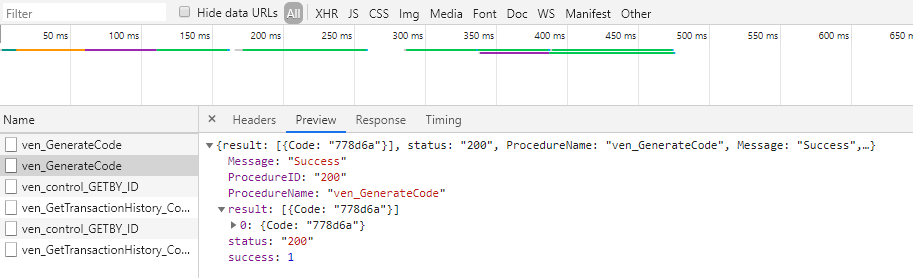
* Information is conditionally rendered (displayed) according with back-end state of acceptances level of user state in predefined containers (empty shells), filled after with response requests from back-end:

FrontEnd Information Exchange

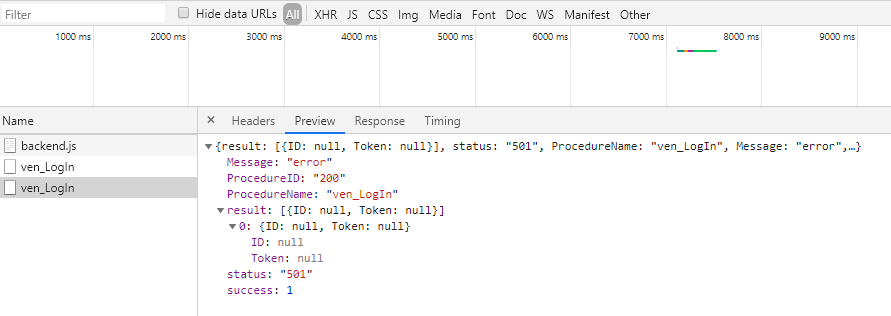


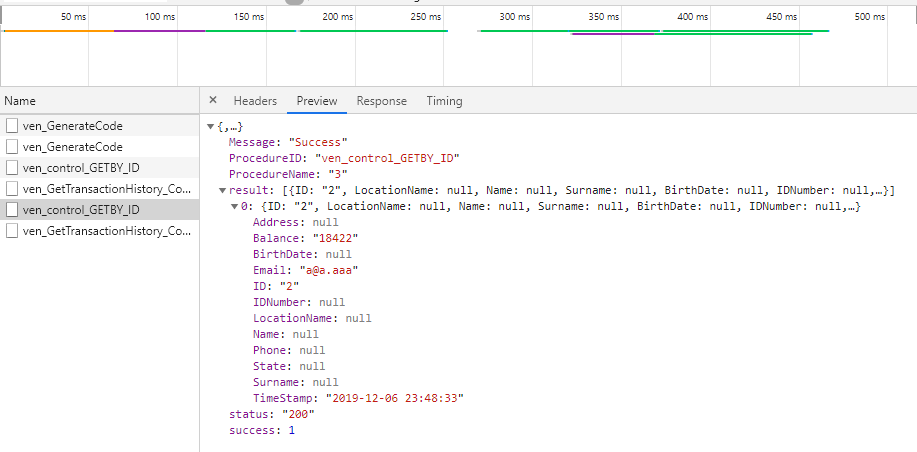
* Sensitive data is not stored locally on device (except user credentials at user own action-request, feature general available at any web-browser level <https://arbitersports.force.com/officials/s/article/How-does-the-Remember-Me-function-work>)
* Personal information about user is obtained by ajax (axios) post requests directly from back-end platform and dynamically loaded into shell components.
* Text Fields are inside local components (with adequate restrictions for properties and behaviors) dynamically created (at every refresh containers and components are destroyed and data is cleared). User receive a new instance of UI elements recreated and rendered in real-time.
* Base principle of information exchange is creating a specific structure object, in dynamic way, locally on user device, taking textInputs values from local state, passing them through a local format validation by regular expression pattern check for acceptance and sent trough https protocol at back-end server for processing. The request replies with a valid response to display into real-time generated components or with a reject message.
* The request procedures do not expose the data sent, but only procedures names and the response contains only coded unique sections of information and/or graphical data to display without direct intelligible correlation within them:



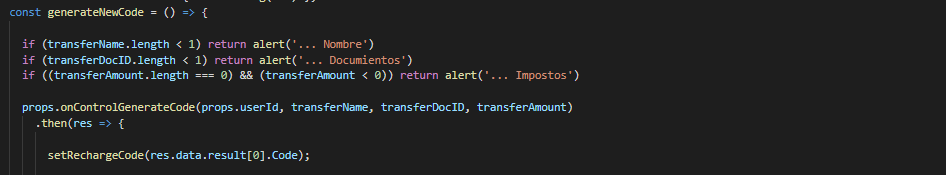


* User private data will be associated with a back-end data-base unique code (or token), and on front-end request side, will be replied the response from server processing in a format containing data, only, in the fields for which the caller has satisfied safe check validation conditions (granted level filtering):





* Transfer of amounts will be handled as transfer of information(fragmented and coded) where the validations checks of crypted data travelling as partial or incomplete information form in a dissociated context trough SSL certified HTTPS will be recomposed in backend by matching algorithms for resulting a success of operation in progress at every current handling.



BackEnd data storage and processing

**System Flow**

* **Login**

To login, user WebApp passes userType, email, password to backend, based on validation of theses on server side, a unique token is generated for that session for specific instance

* **1.2 Sign Up**

Inputs for Control User Signup: **Ubicación (nombre), Nombre, Apellido, Fecha nacimento, Num nacional identidad, Dirección, Estado, Teléfono, Email, Password;**

Inputs for Client User Signup: **Nombre, Apellido, Fecha nacimento, Num nacional identidad, Dirección, Estado, Teléfono, Email, Password**

Inputs for Comercio User Signup**: Razón social/Nombre legal, Dirección, Estado, Nombre, Apellido, Teléfono, Email, Password**

* **1.3 Client to Comercio flow**

Add Payment Request

Each comercio registration receive a unique comercio id that will be used (once comercio login credentials are accepted), along with QR code reader instance resulted data. After scanning client’s unique QR generated pictogram, representing user identification unique number (generated in client section) and the inserted the amount comercio logged in operator can send the request to make payment transfer.

To add payment request WebApp passes clientId (sender), comercioId (receiver) and amount to be transferred based on validation of these on server side .

A unique transfer request id (transaction id) is generated and saved in database for further references with transaction status 'pending' and a notification using FCM push notification service is sent to user to approve/reject that transaction

Approve/Reject Payment Transfer Request

To approve transfer WebApp passes requestId to be validated and acknowledged by the user in his own WebApp section interface. That is generated in above method and based on validation (client approval or ignorance) of this request, on server will be updated the corresponding user and comercio balance .

When client accepts, the dates on server are updated and transaction status will get updated also to approve/reject.

* **1.4 Control to Client flow**

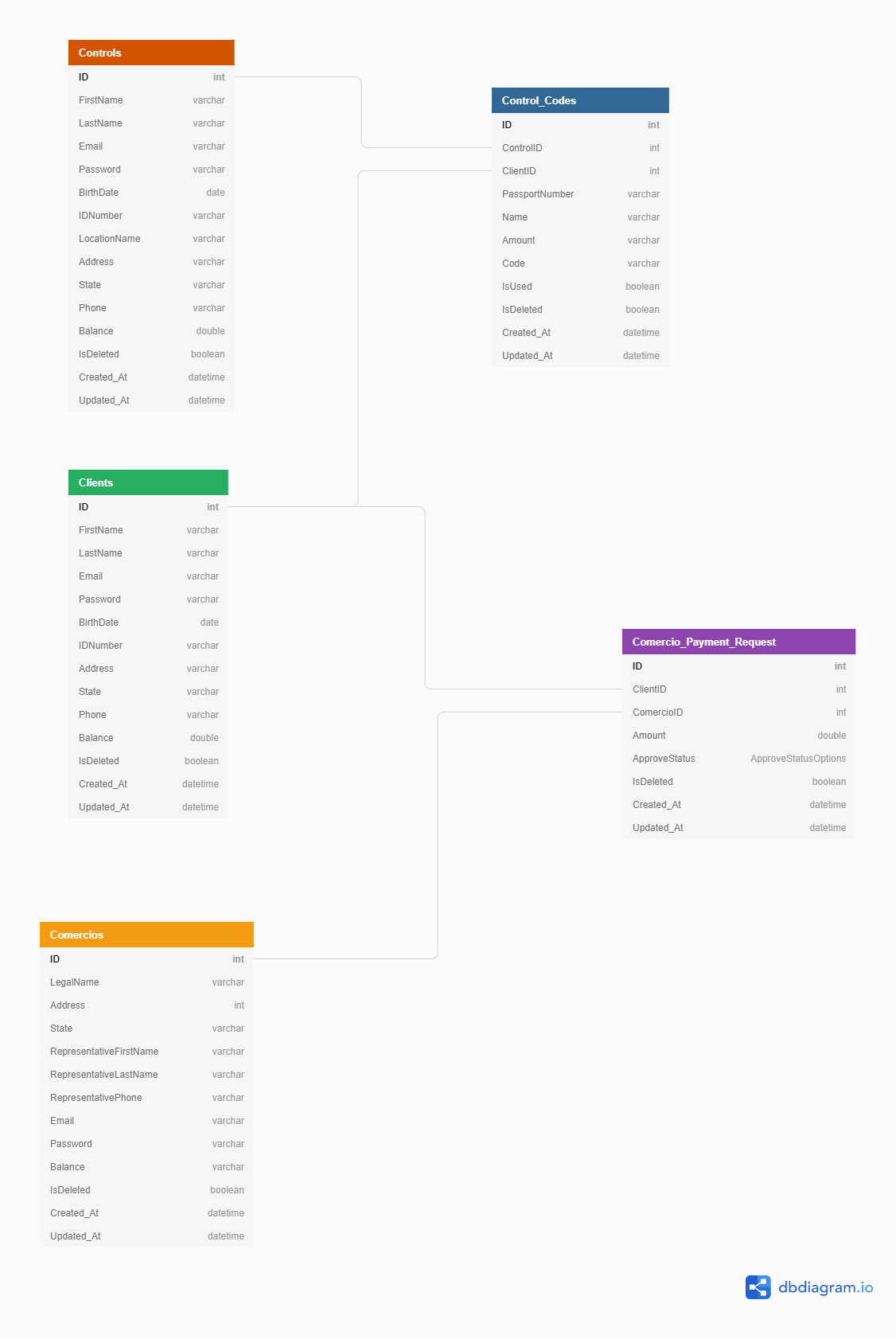
**Note**: Each Control user will have its own UNIQUE Control id

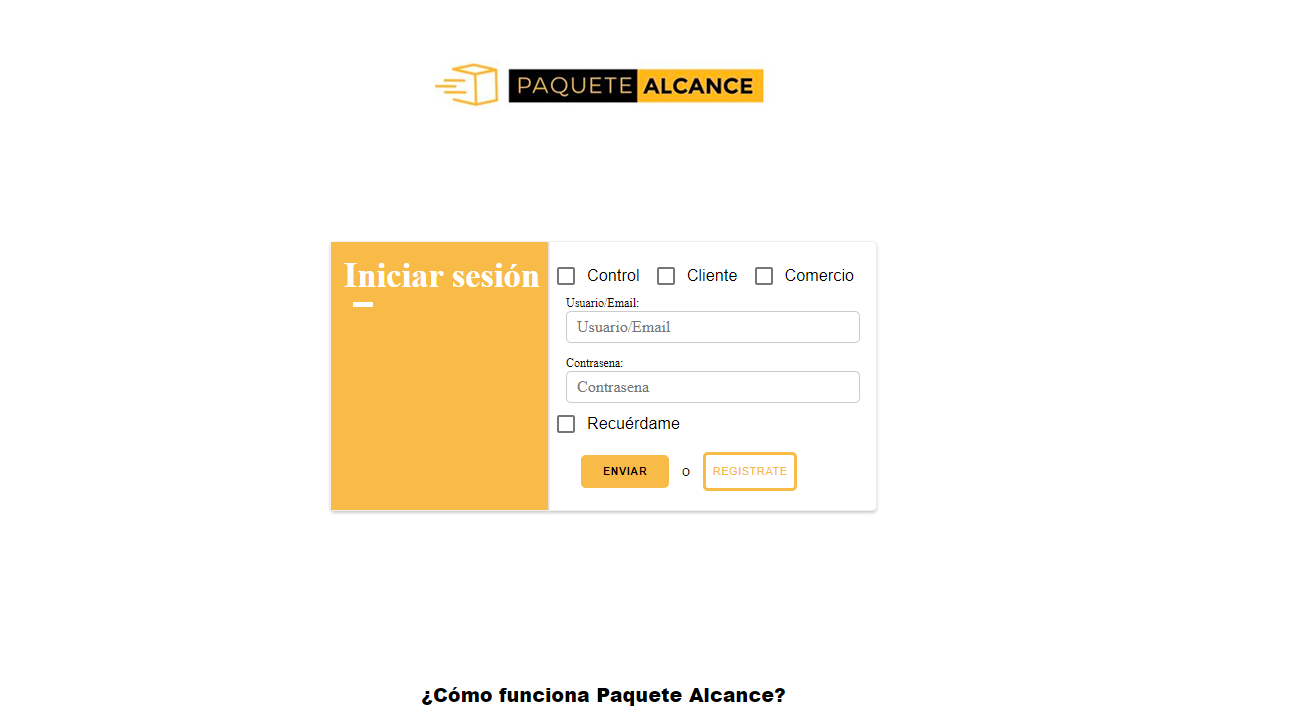
Whenever Control will get new request of transfer, Control User should collect data from requester as follows:

Name

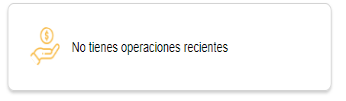
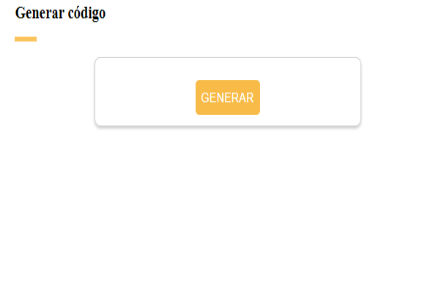
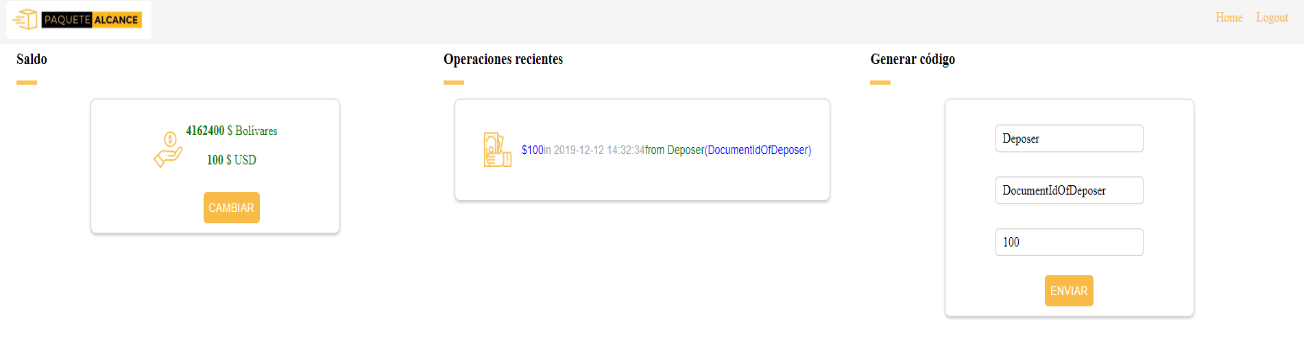
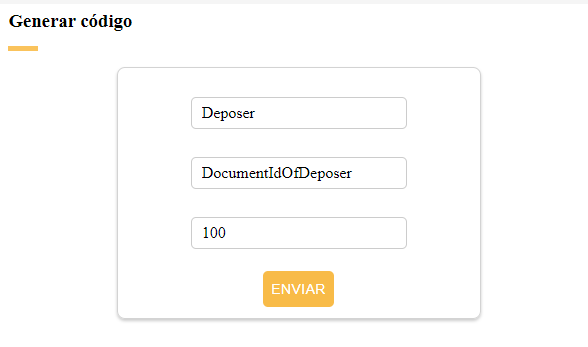
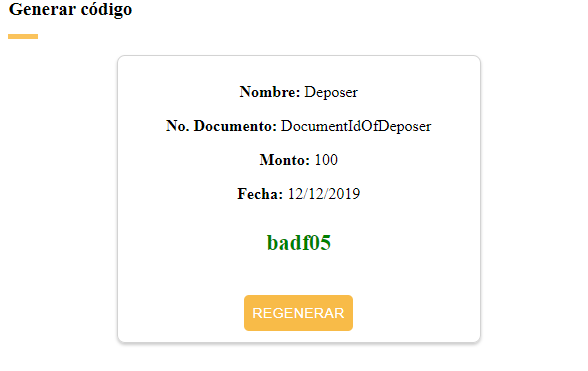
Passport Number

Amount to be transferred

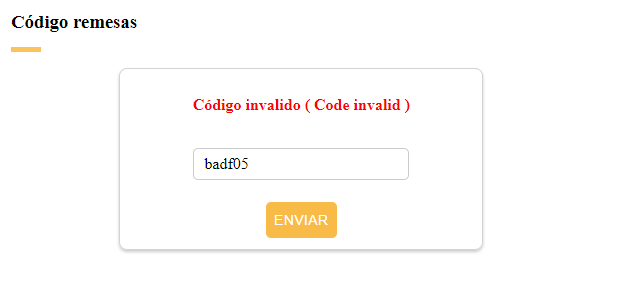
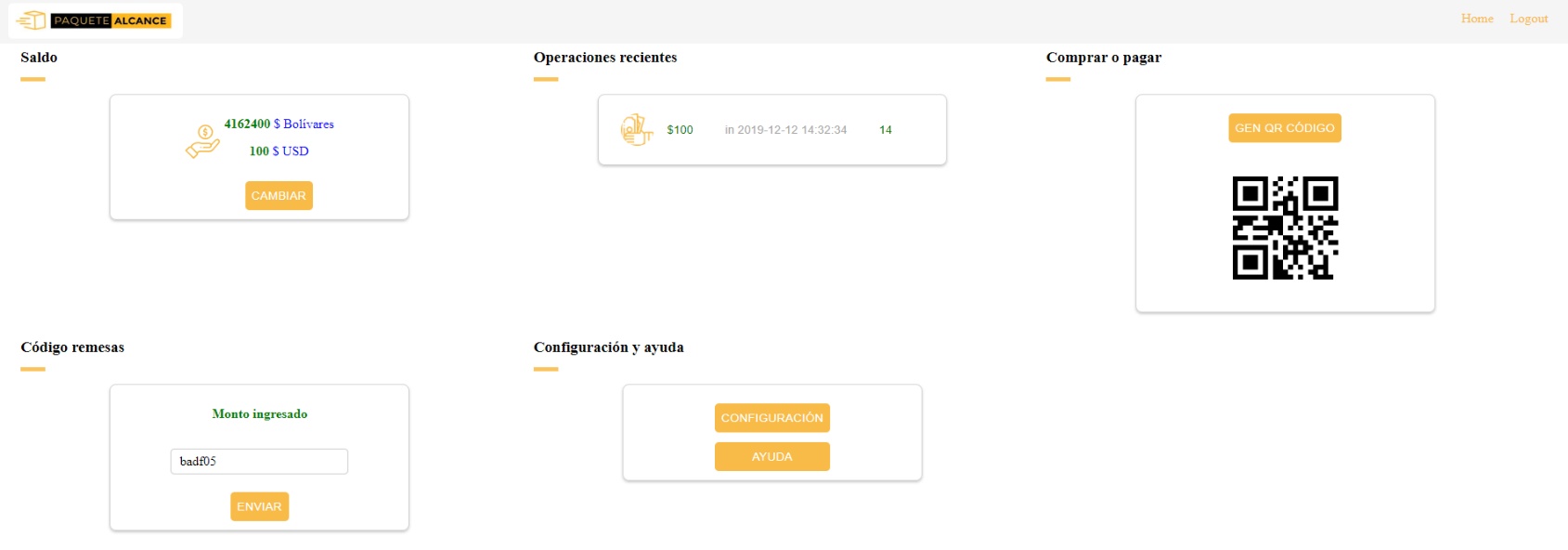
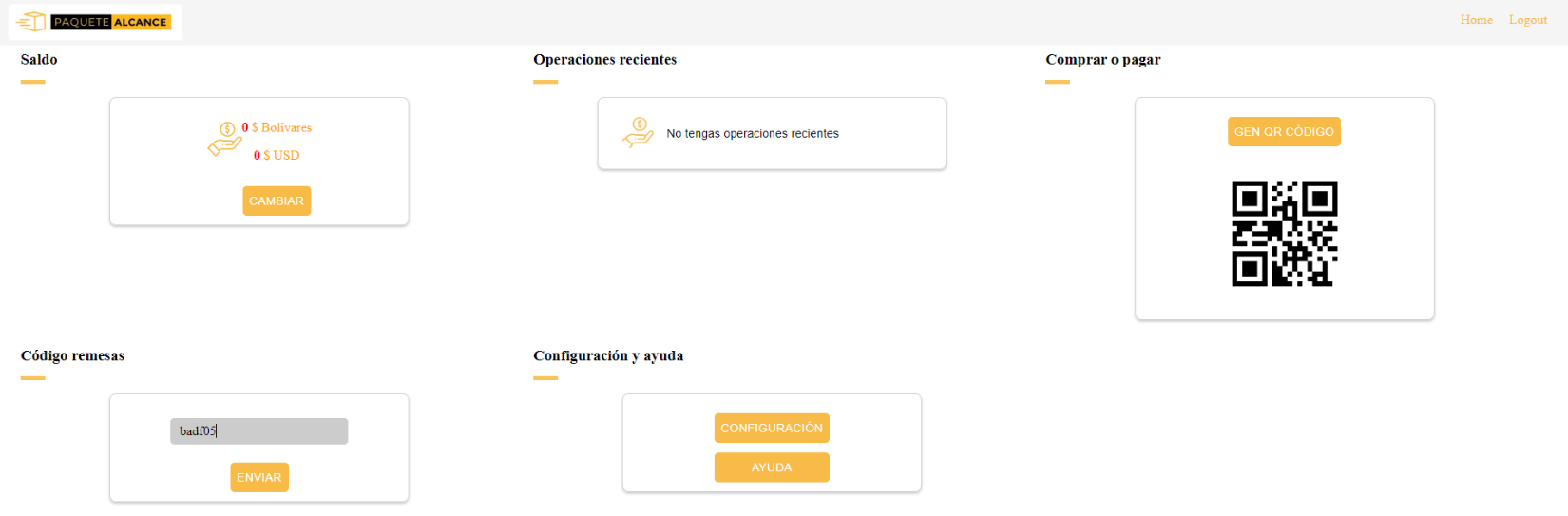




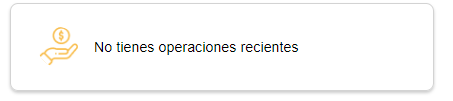
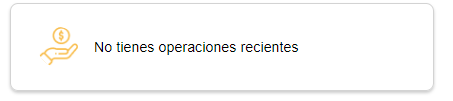
CONTROL SCREEN



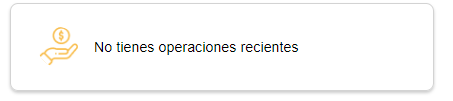
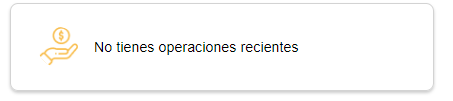
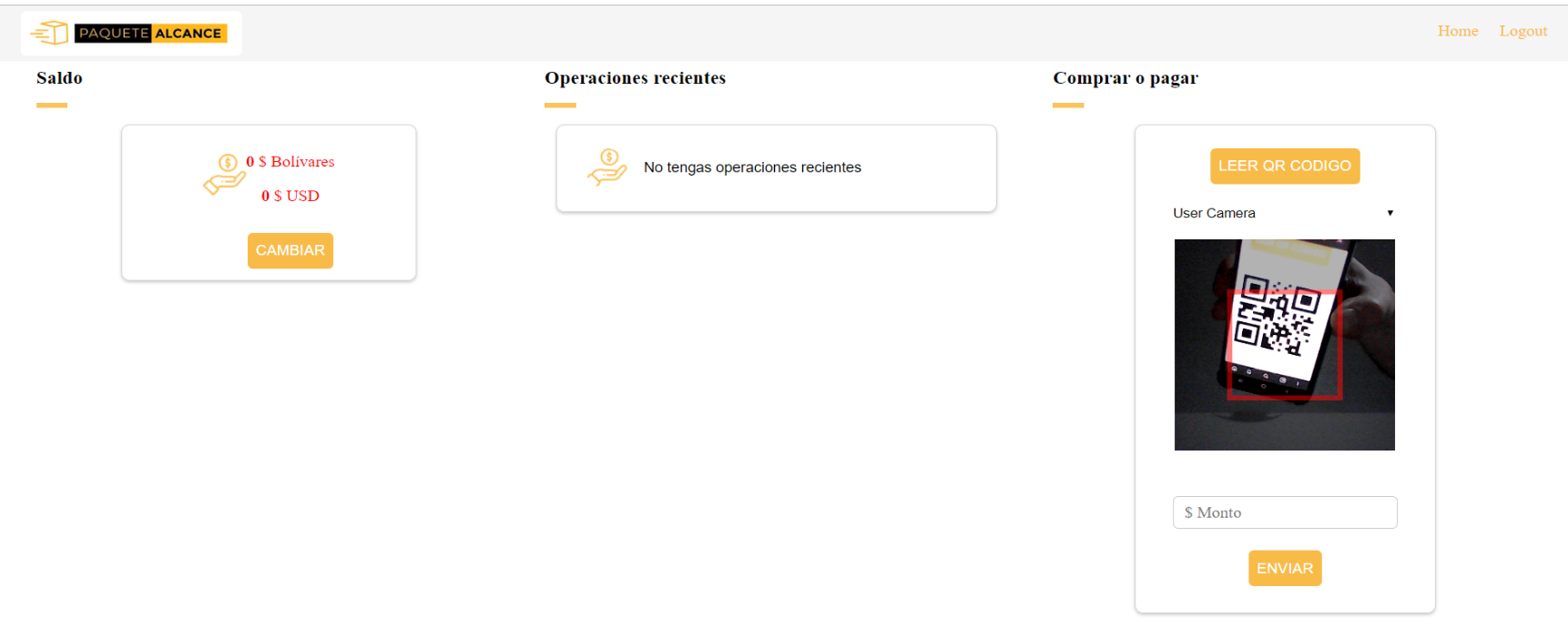
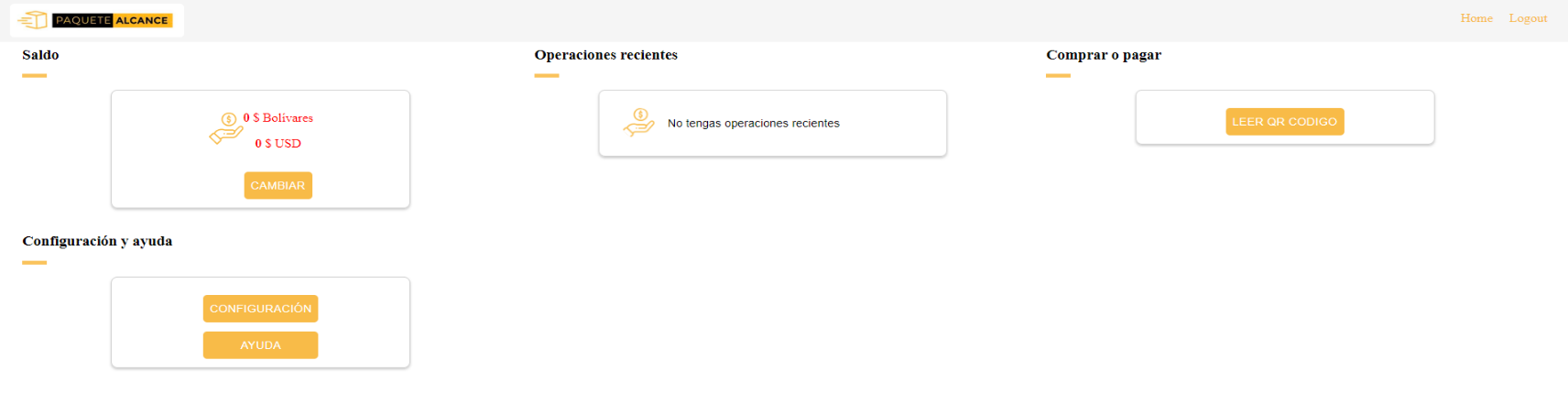
CLIENT SCREEN



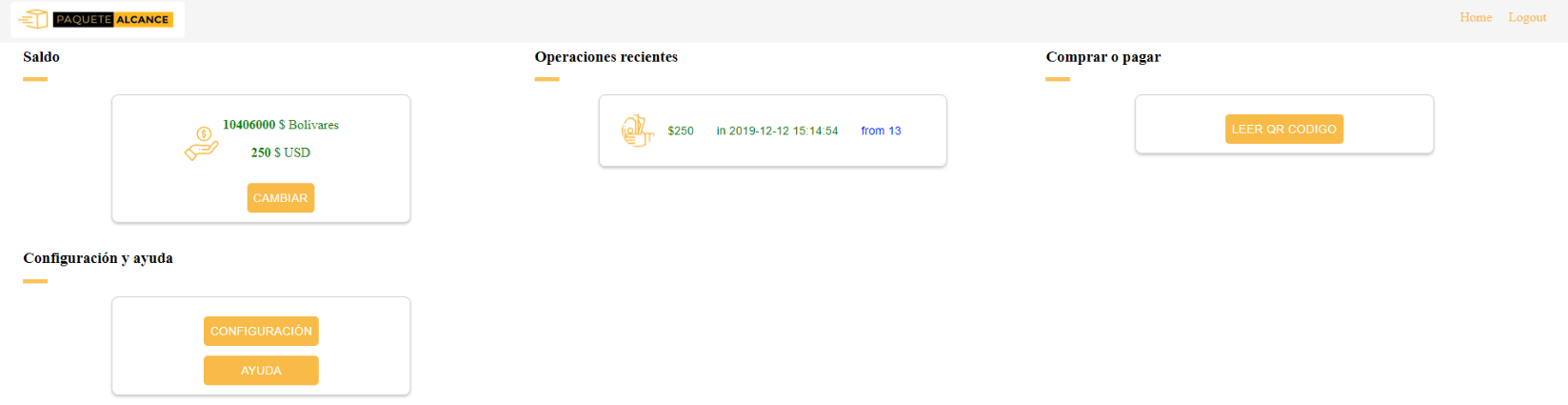
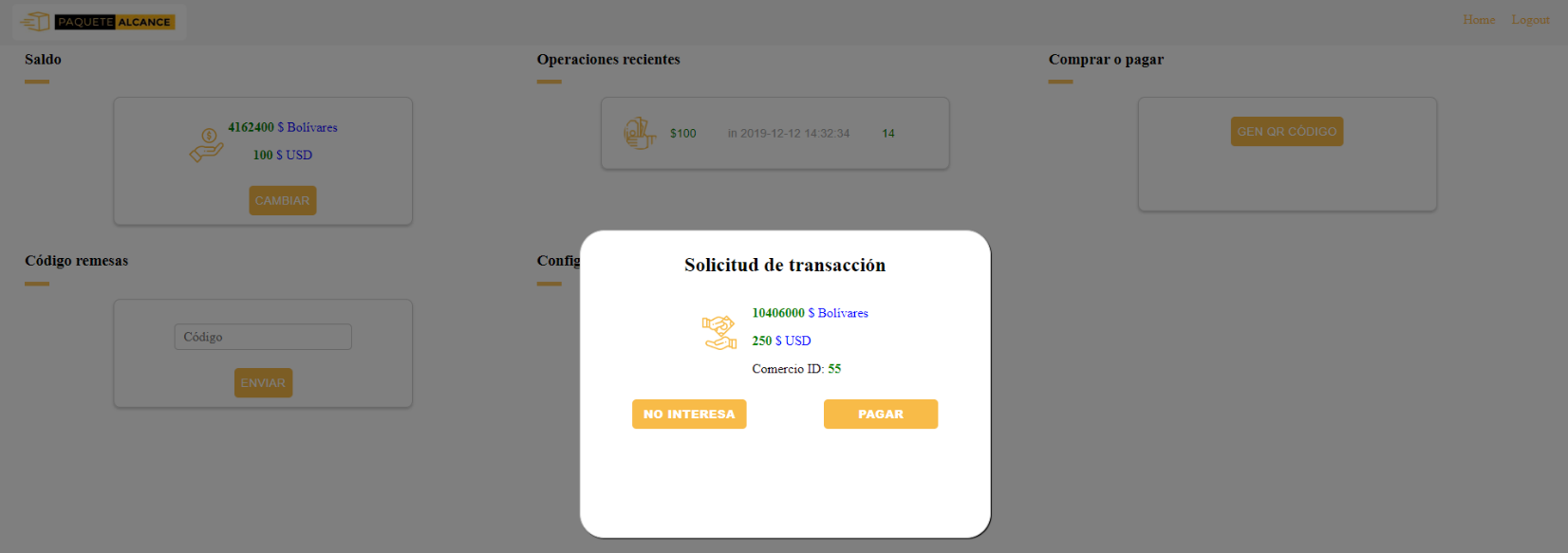
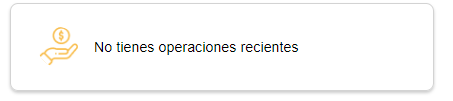
…



COMERCIO SCREEN



CLIENT SCREEN



COMERCIO SCREEN