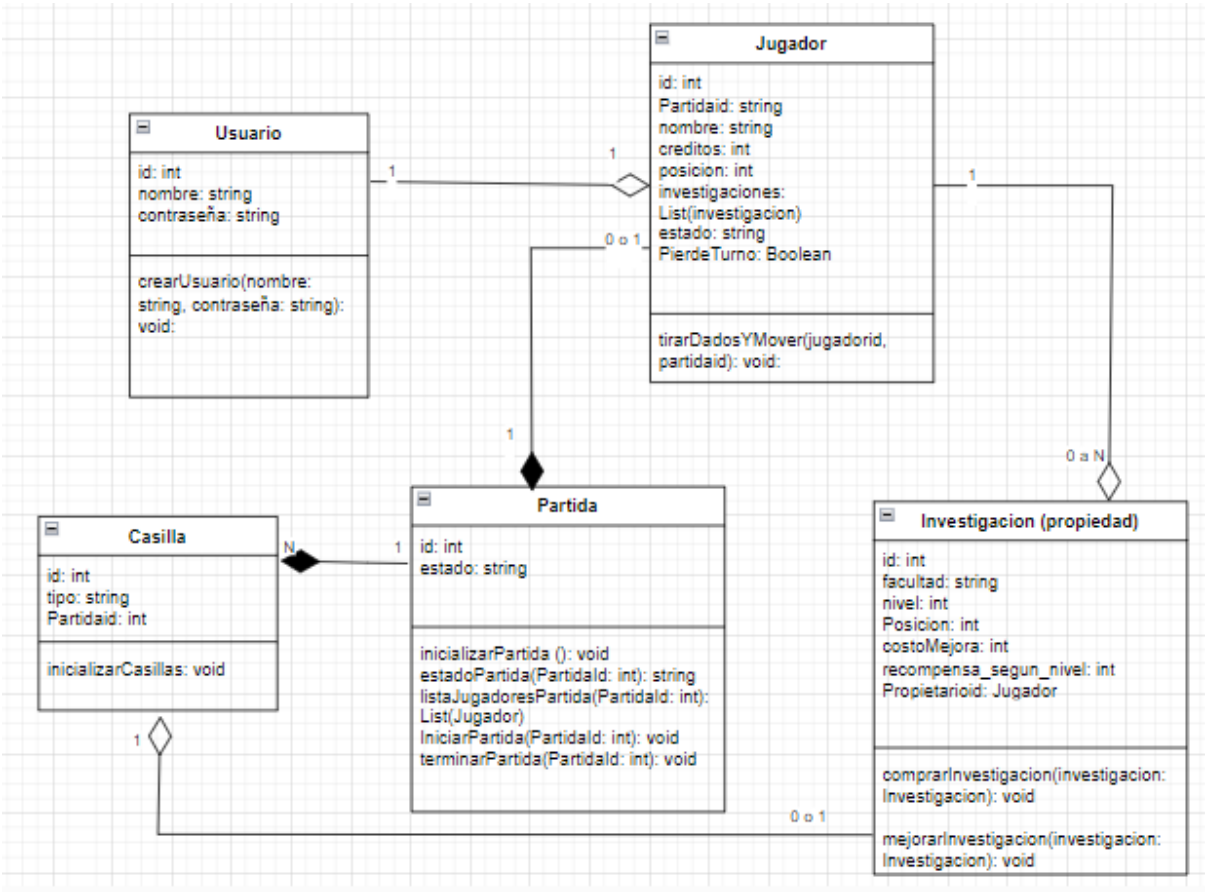
1. **Diagrama UML**



1. **Usuario**
    a. **Atributos:**
        i. id: int: Entero único como identificador de Usuario.
        ii. Email: string: Email para poder registrarse
        iii. nombre: string: Nombre del jugador.
        iv. constraseña: string: contraseña con la que entrará el usuario.
    b. **Métodos:**
        i. CrearUsuario(nombre, , contraseña): Crea el usuario con el nombre, correo y contraseña entregado
        ii. LoginUsuario(correo,contraseña): Inicia sesion con su correo y contraseña
        iii. ObtenerUsuario: Se devuelven los datos del usuario a partir del token
        iv.

2. **Jugador**
    a. **Atributos:**
        i. id: int: Entero único como identificador de jugador.

ii. UsuarioId: int: Usuario a que usuario pertenece
iii. PartidaId: string: Partida a la que es perteneciente
iv. nombre: string: Nombre del jugador.
v. creditos: int: número de creditos que tiene el jugador.
vi. posicion: int: número de casilla donde esta jugador.
vii. investigaciones: List(Investigacion): Lista de investigaciones que tiene el jugador.
viii. estado: string: Estado actual jugador (activo, bancarrota).
ix. Turno: int: Define el turno en el que le toca en la partida
x. PierdeTurno: Boolean: Indica si pierde el siguiente turno

b. **Métodos:**
i. tirarDadosYMover(jugadorId, partidaId): void: Mueve al jugador el número de posiciones que entregue el dado que tiró el jugador.
ii. CrearJugador(IdUsuario, partida): Se crea un jugador para el usuario en cierta partida
iii. ObtenerEstadoJugador(jugadorID): se obtiene el estado del jugador en una partida
iv. CambiarCredito(idJugador): Se cambia el credito de un jugador

3. **Investigacion (Propiedad)**
a. **Atributos:**
i. id: int: Número único de identificador de la propiedad.
ii. facultad: string: Facultad a la que pertenece la investigación.
iii. nivel: int: Nivel de la investigación (1: Trabajo Semestral, 2: Trabajo Anual, 3: Proyecto de Título).
iv. Posicion: int: posicion en el tablero de la investigación
v. costoMejora: int: Costo en número (créditos) para avanzar de nivel en la investigación.
vi. recompensa_segun_nivel: int: Cuantos creditos gana segun el nivel
vii. PropietarioId: Jugador: El jugador que es dueña de la investigación.
viii. PartidaId: referencia a que partida pertenece la investigacion

b. **Métodos:**
i. comprarInvestigacion(investigacion: Investigacion): void: Comprar investigación.
ii. mejorarInvestigacion(investigacion: Investigacion): void: Mejorar investigación.

4. **Casilla**
a. **Atributos:**
i. id: int: Número identificador único de la casilla.

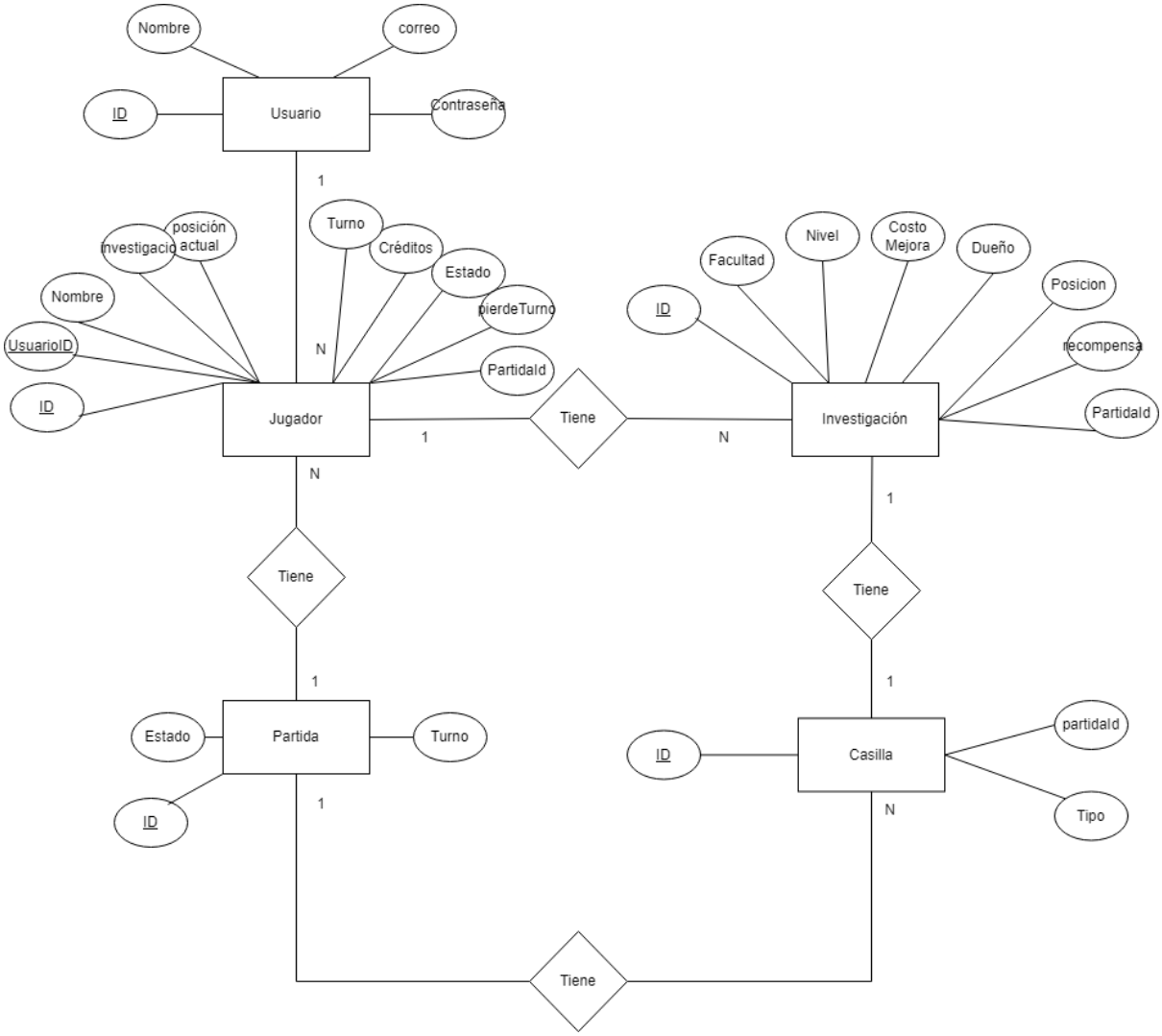
ii. tipo: string: Tipo de casilla (inicio, investigación, fortuita, secretaría académica, neutra, váyase a secretaría académica).
iii. PartidaId: integer: referencia a la partida que pertenece

b. **Métodos:**
i. InicializarCasillas: void: Insertar a la tabla las casillas de la partida

5. **Partida**
a. **Atributos:**
i. Id: Integer: identificador de la partida
ii. estado: string: Estado de la partida (todavía no comienza, en curso, finalizado).
iii. TurnoActual: int: representa a que jugador le toca jugar

b. **Métodos:**
i. inicializarPartida (): void: Iniciar una nueva partida.
ii. estadoPartida(PartidaId): Verificar estado de la partida.
iii. listaJugadoresPartida(PartidaId): devuelve la lista de jugadores de la partida
iv. IniciarPartida(partidaId): Se inicia la partida
v. terminarPartida(PartidaId): da por terminada la partida

2. Diagrama ER



1. **Entidad Usuario**: La identidad usuario tiene como llave primaria su id como identificador unico, tambien posee nombre, correo y contraseña.

2. **Entidad Jugador**: La entidad Jugador tiene como llave primaria el id, la cual es única e identifica a cada jugador en el juego. Además, cuenta con una llave foráneas: Partida_Id y usuarioID. La llave foránea partidaId indica a que partida pertenece y usuarioID a que usuario pertenece. Los demás atributos son Nombre, que almacena el nombre del jugador, Créditos, que registra los créditos disponibles, Estado, que indica si el jugador está activo o inactivo en la partida, turno indica cuando le toca jugar,pierdeTurno indica si pierde el siguiente turno y posición actual, la cual indica la casilla en la que esta.

3. **Entidad Investigación**: La entidad Investigación tiene como llave primaria el atributo id, que permite identificar de manera única cada investigación en el

juego, y con partidaId para saber a que juego pertenece. Cuenta también con un atributo llamado posicion, que indica a que casilla corresponde. Los atributos adicionales de la investigación son Facultad, que define el área de especialización de la investigación, Nivel, que especifica el progreso alcanzado, Costo Mejora, que almacena el costo requerido para incrementar el nivel de la investigación y recompensa, que indica cuantos créditos recibirá el dueño cuando otro jugador cae ahí.

4. **Entidad Partida**: La entidad Partida tiene como llave primaria el atributo id, el cual identifica de manera única cada partida en el juego. Además, contiene un atributo llamado Estado, que almacena el estado actual de la partida (en curso, finalizada, etc.) y un atributo turno, que determina a que jugador le toca.

5. **Entidad Casilla**: La entidad Casilla tiene como llave primaria el id, que identifica a cada casilla de forma única dentro del tablero. Esta entidad cuenta con una llave foránea llamada partidaId, que indica a cuál partida pertenece la casilla. El otro atributo es Tipo, que define la función de la casilla (normal, carta sorpresa, cárcel, etc.).

3. Documentación para testeo.

Al igual que la entrega pasada, se utilizó Postman para probar los endpoints programados, pero producto de la implementación de los tokens y las sesiones dentro del código, hay algunos endpoints que resultan más tediosos de probar por medio de Postman, y resultan mucho más cómodos de testear a través de la página misma. Pese a esto, se indicará de igual manera cuales son los testeos más faciles de realizar en la página o en Postman, y se indicará como realizar su testeo a través de Postman.

También he de mencionar que, de acuerdo con lo conversado en la reunión de la entrega pasada, se cambió todo el proyecto a “type: commonJS”, y desde entonces no volvimos a tener problemas. Mencionar también que para hacer este cambio fue usado chatgpt para hacerlo más rápido.

Dentro de la carpeta de entrega 2 que se encuentra en el backend, se encuentra un un archivo llamado Test entrega 3 web.postman_collection, el cual es la carpeta con los testeos que se realizaron para probar los endpoints.

Para importar la carpeta, una vez en postman, se tiene que apretar el botón de “Import” ubicado a la derecha de donde dice “My Workspace”, y seleccionar el archivo Test entrega 3 web.postman_collection.

Antes de comenzar con los testeos, se tiene que hacer el comando de “sudo service postgresql start” en ubuntuu, y luego correr yarn dev en la ubicación del proyecto del backend. De igual manera, hay que correr yarn dev en la ubicación del proyecto del frontend.

Considerar también que para el correcto funcionamiento de la pagina, hay que crear un archivo .env tanto en el front como en el back. El .env del back tiene que contener la siguiente información:

DB_USERNAME =
DB_PASSWORD =
DB_NAME = proyecto_web //cambiable a su base de datos
DB_HOST = "localhost"
JSW_SECRET =jwt_secret

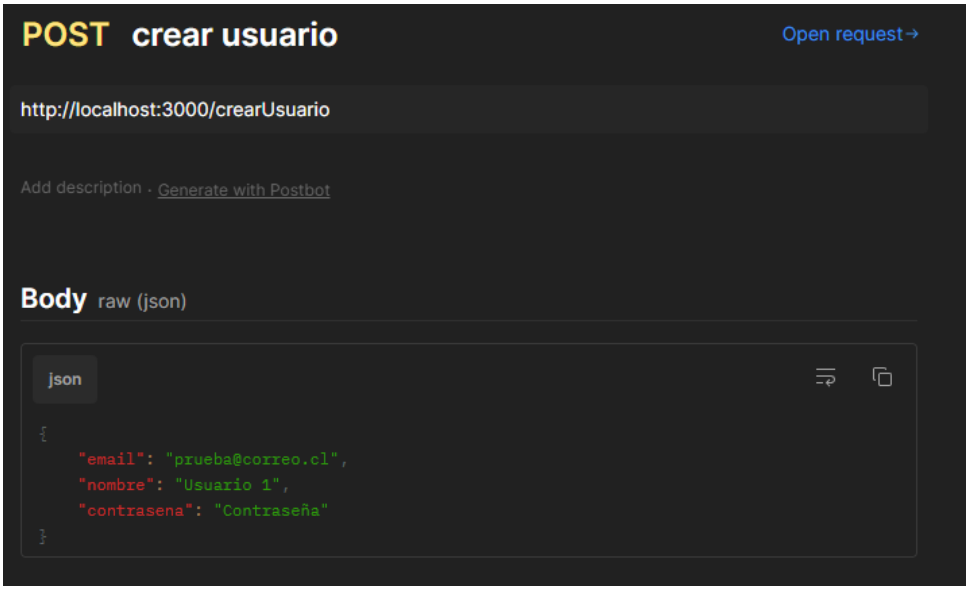
Mientras que él .env del front tiene que contener lo siguiente:

VITE_BACKEND_URL = "http://localhost:3000"

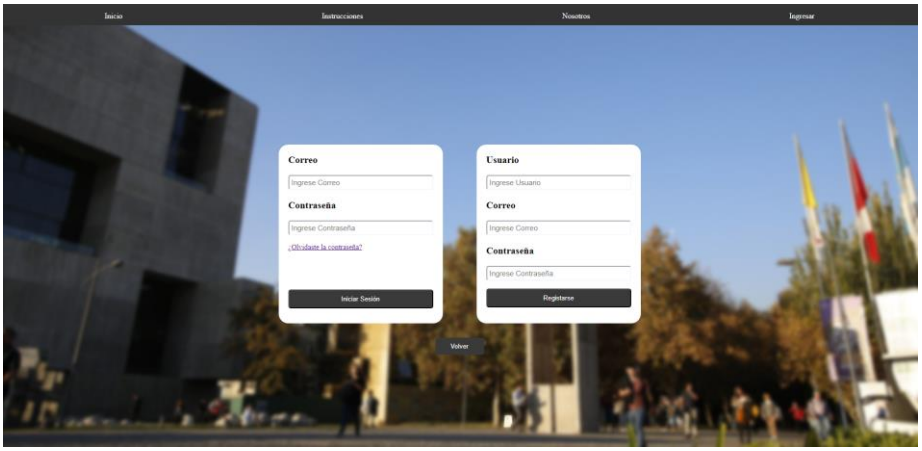
Finalmente, tener en cuenta que fueron instaladas todas las dependencias necesarias indicadas en las capsulas del proyecto, por lo que en caso de no tener alguna instalada puede causar problemas. Además, dado que fueron modificadas algunas estructuras de ciertas BDD, se recomienda hacer DROP TABLE en caso de que se encuentren creadas de la entrega pasada.

Una vez ya corriendo el yarn dev, podemos utilizar postman para testear los endpoints.

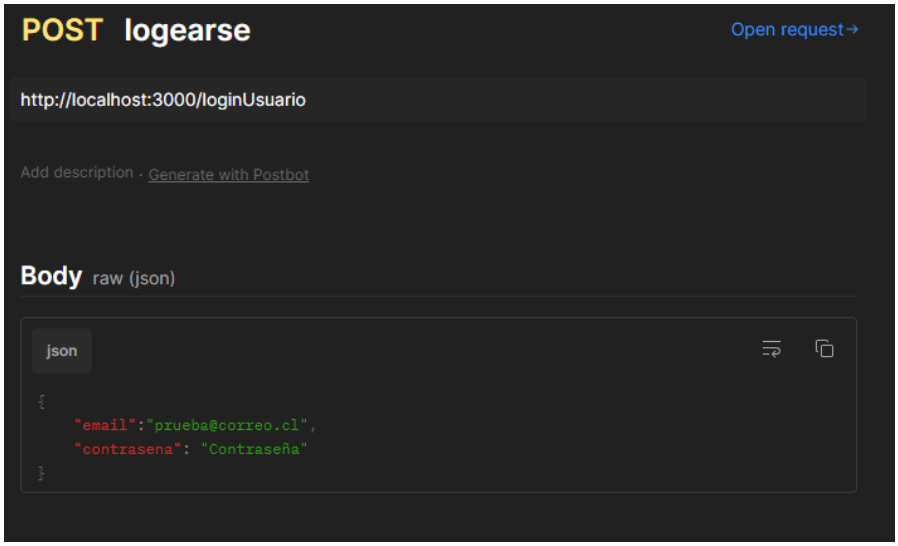
1. Crear usuarios



Para probar el primer endpoint, se utiliza el post crear usuarios. Este endpoint también es testeable a través de la página como se puede ver en la siguiente foto, ingresando el nombre de usuario, el correo y la contraseña.

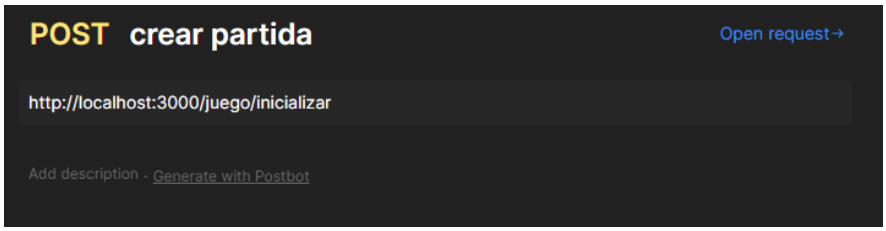


2. Login



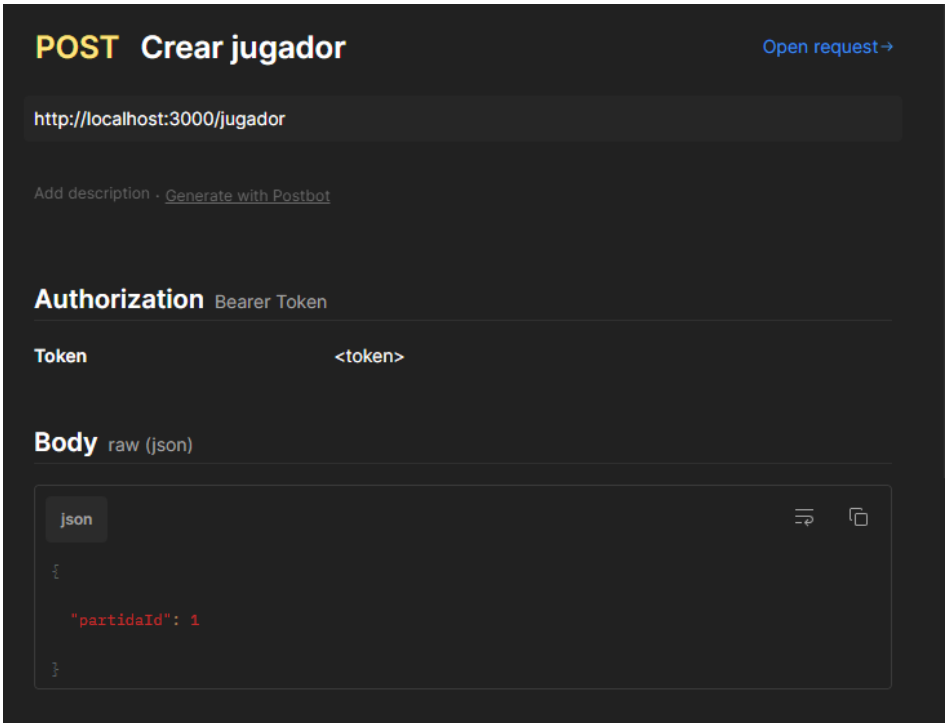
Para testear el endopint de login, se puede testear tanto desde Postman como desde la página. Se tiene que entregar el correo y la contraseña .

3. Crear partida



Para poder testear el tercer endpoint de crear partida, solo se puede hacer desde Postman, ya que todavía no tenemos implementado la lógica de cuándo se va a crear una partida en el front. Solo se tiene que ejecutar, ya que no recibe nada a través del body.

4. Crear jugador



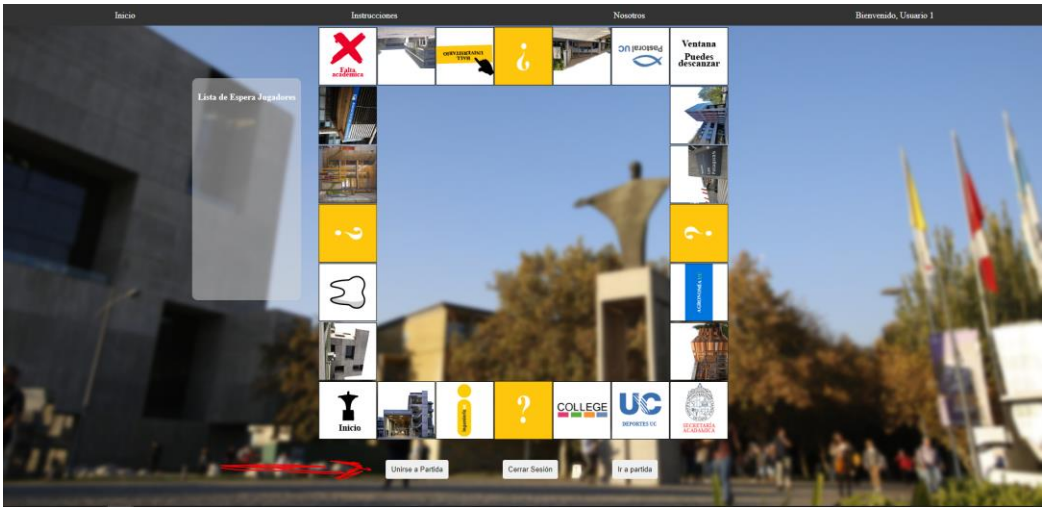
Este es uno de los endpoints que es más fácil de probar en el front, ya que para poder crear un jugador, tiene que haber un usuario logueado, y esto se comprueba a través del uso de tokens. De igual manera se puede testear desde Postman, pero tiene que asegurarse de tener el token obtenido al registrarse o al loguearse con los test pasados.

Es importante que se cree primero un usuario y una partida, ya que los jugadores pertenecen a un usuario y son parte de una partida. El id del usuario se obtiene a partir del token, y el id de la partida a la que se quiere unir se pasa a través del body.

Si se intenta crea el 4to jugador asociado a la partida, esta comienza automáticamente. Y en caso de intentar crear un jugador más a esta partida, dará el error de máximo 4 jugadores.

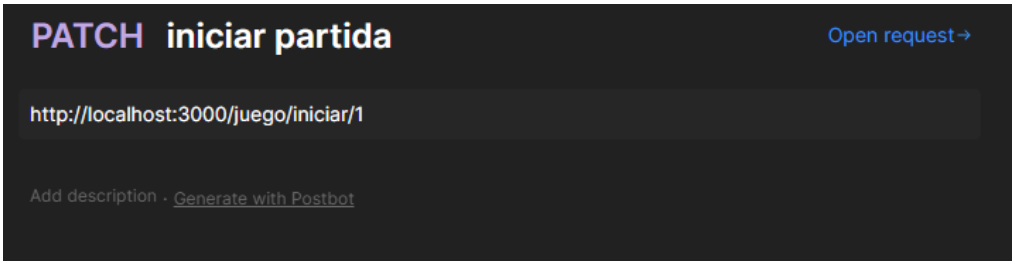
Adicionalmente, hay que tener en cuenta que como todavía no se crean partidas en el front, tiene que ser creada una partida a través de Postman antes de probar este endpoint. Y en el front se encuentra hardcodeado para intentar conectarse a la partida de id=1 momentáneamente.

En la siguiente foto se indica como testear el endpoint a través del front.



Al momento de clickear el botón de unirse a la partida, saltará un aviso si se logró o no conectar, y en caso de que si, será redireccionado a la vista de partida (momentáneo)

5. Iniciar partida

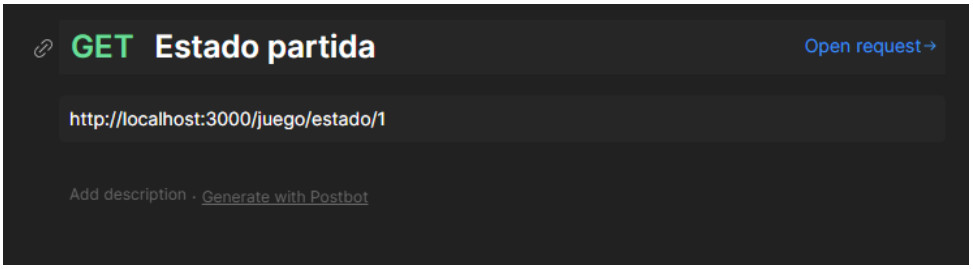


Este endpoint solo puede ser probado desde Postman, ya que queremos que en el futuro sea ejecutado por un administrador, pero primero tenemos que implementar la lógica de partida al front, y luego la de administrador.

Para iniciar la partida, solo basta con ejecutar el endpoint. Verifica que la partida que se quiere iniciar exista, que no haya sido empezada ya y que no haya sido terminada previamente. En caso contrario, da el error correspondiente

Cabe destacar que la partida no será iniciada si cuenta con menos de 2 jugadores asociados. El orden de turnos es en orden de llegada

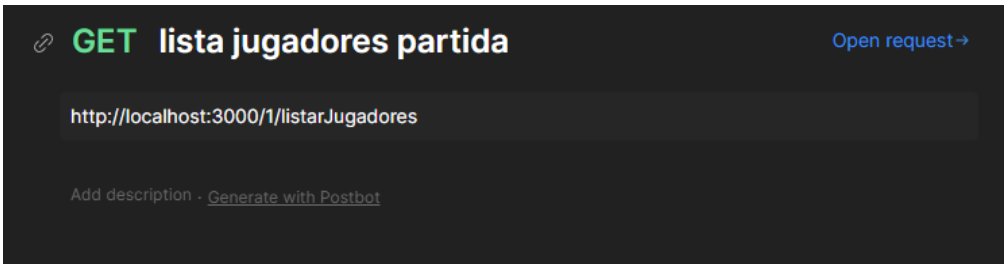
6. Estado partida



Nuevamente, este endpoint solo se puede probar dentro de Postman, ya que todavía no lo implementamos de manera util en el front.

Sirve para consultar el estado de la partida. Se tiene que entregar el id de la partida a consultar en la petición y no en el body. En caso de no existir el id entregará el error de que no existe esa partida

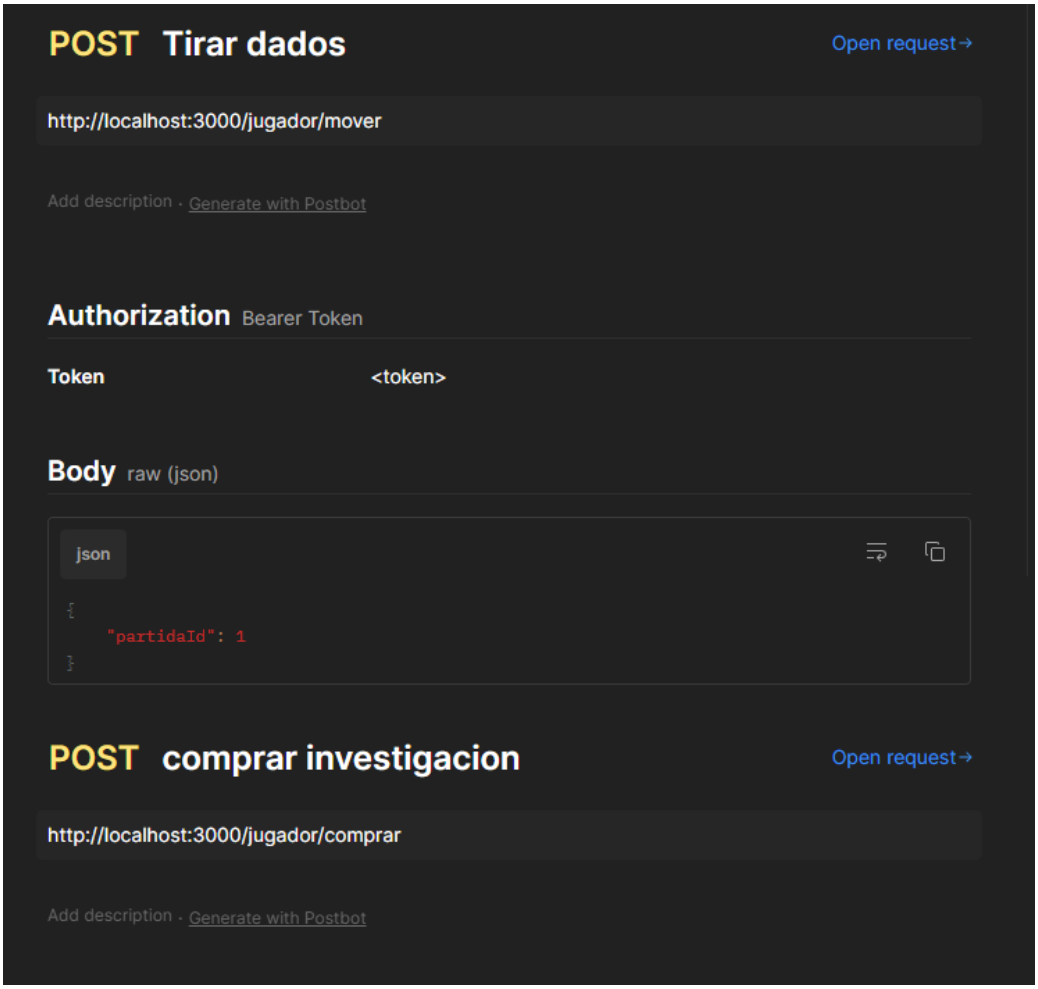
7. Lista jugadores



Si bien este endpoint es utilizado actualmente el en front, no hay una manera directa de probarlo, ya que es utilizado a la hora de obtener la información de los jugadores cuando se está jugando para actualizar la posición de los jugadores en el tablero del front.

Este endpoint es para consultar la lista de los jugadores de cierta partida. Nuevamente el id de partida tiene que ser entregado en la consulta/url y no en el body. En caso de no existir el id entregará el error de que no existe esa partida

8. Tirar dados



Este es uno de los endpoints que nuevamente vale la pena probar a través de la página y no con Postman, ya que hay que ingresar el token del jugador que le toca.

En endpoint es uno de los más importantes del programa, ya que con este los jugadores juegan. Se tiene que entregar en el body el id de la partida, ya que con el token se obtiene el id del jugador. Verifica que el que está jugando es el jugador que le toca, sino tira el error correspondiente. También se verifica que la partida este inicializada, ya que en caso contrario no dejará

El jugador tirará el dado y avanzará por el tablero actualizando su posición. Este puede caer en cualquier tipo de casillas.

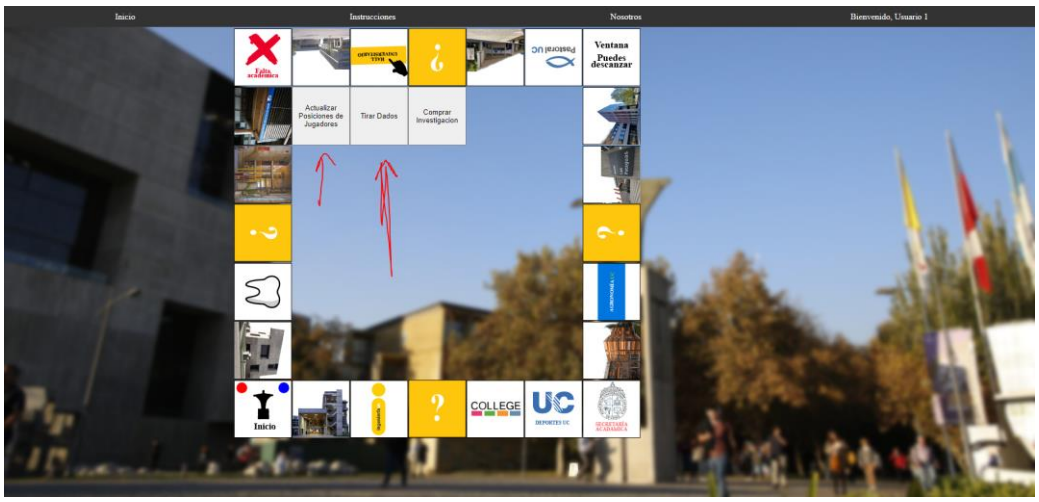
En caso de caer ventana, inicio o secretaría académica, no ocurrirá nada. En caso de caer en falta a la integridad, se va a la casilla de secretaría académica y pierde el turno. En caso de caer en la casilla de fortuna, se escoge una carta al azar y puede ganar o perder créditos, o irse a secretaría académica y perder el punto. En caso de caer en una casilla de investigación, el jugador tendrá la opcion de comprar una investigación, o si ya es dueño de esa investigación, mejorarla. En

caso de caer en la investigación y no ser dueño de esta, al jugador se le descuentan la cantidad de créditos de recompensa de la investigación, y si no tiene el monto suficiente, se le descuenta lo que pueda y cae en bancarrota, es decir, pierde.

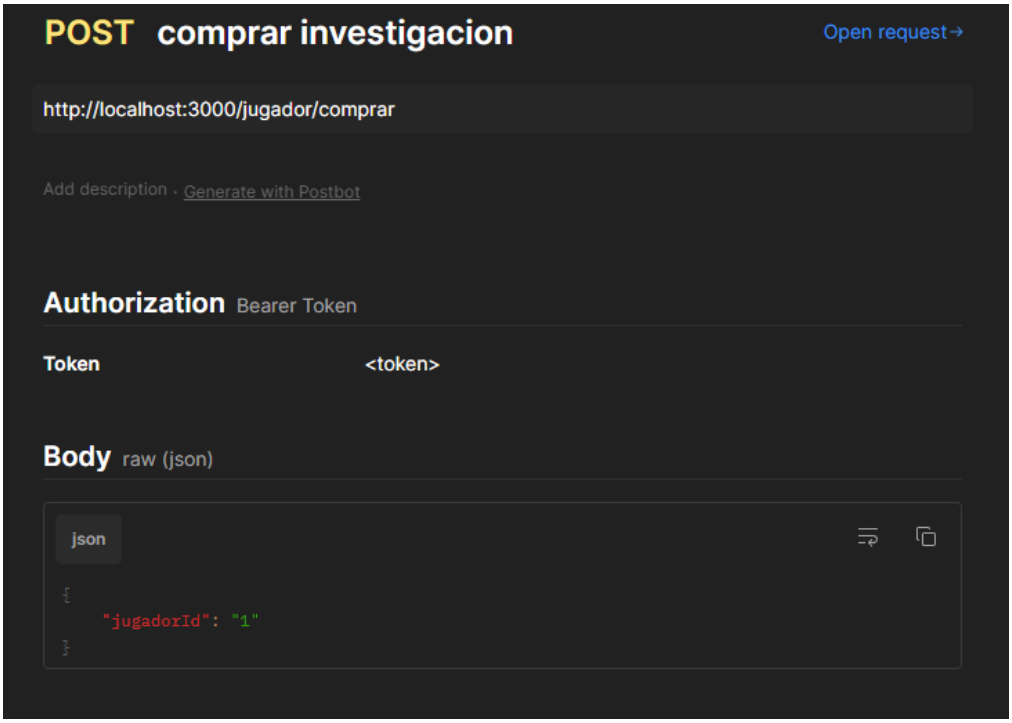
Finalmente, en cada lanzada/turno verifica cuantos jugadores quedan activos, y en caso de quedar solo uno, se da como terminada la partida y como ganador al último jugador.

Para poder testearlo de mejor manera se puede realizar a través del front. Una vez se haya iniciado la partida, ya sea por que se unieron 4 jugadores o porque se inició con el endpoint, uno puede ver en la vista de partidas el botón de tirar dados. Tirar los dados tendrá efecto solo cuando es tu turno, y una vez clickeado, se actualizará la posición del jugador tanto en el front como en el back, y se podrá ver en que casilla quedó la ficha. Dado que no se implementó websockets, si el jugador 2 quiere ver los cambios de posición una vez jugados, tiene que apretar el botón de “Actualizar Posiciones de Jugadores “.

Finalmente, el jugador 1 será representado por una ficha color rojo, el 2 por azul, el 3 por verde y el 4 por amarillo. El orden de los jugadores se determina en el orden que se unieron.



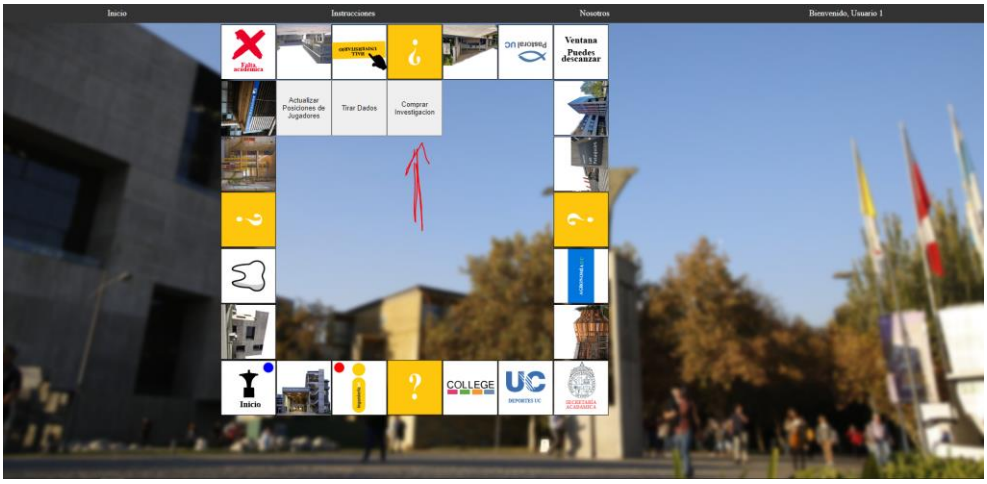
9. Comprar investigación



Nuevamente este endpoint es más facil de probar en el front, y se tiene que realizar mientras se está jugando/probando el endpoint de tirar dados. En caso de querer probarlo en Postman, se debe tener cuidado con el token, ya que tiene que ser el usuario que tiene un jugador que se encuentra en una casilla habilitada para comprar.

Para este endpoint, se requiere el id de la partida, ya que el jugador y s posición se obtiene a través del token. Se verifica que sea una casilla de investigación, y que no caso de serlo, que no tenga dueño. Se verifica si el jugador tiene los créditos necesarios para comprarla, y en caso de que si, la adquiere, descontandole los créditos respectivos. En caso de no tener los créditos necesarios, lanza el error correspondiente.

Para probarlo en el frontend, se tiene que apretar el boton de “Comprar Investigación” y se ejecutará en caso de cumplir con las condiciones. Todavía no hay nada que demuestre que el usuario compró la investigación en el front, pero si se mira la consola, se puede comprobar que se realizó correctamente.

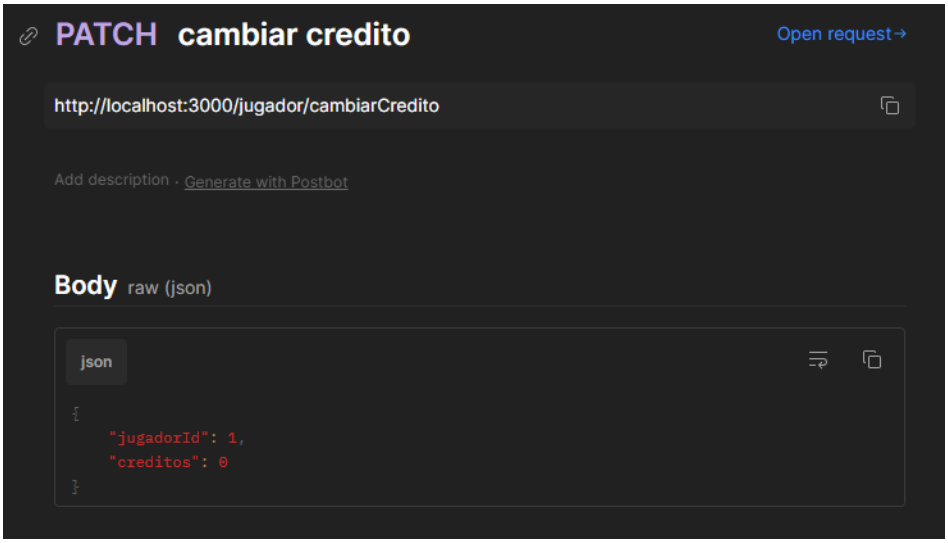


10. Mejorar investigación

Este endpoint solo puede ser probado en Postman, ya que aún no es implementado su funcionamiento en el frontend por tiempo, pero utilizaría la misma lógica que endpoint pasado de comprar investigación

En este endpoint se verifica 4 cosas, que la casilla en la que se encuentre sea una investigación, que el jugador sea dueño de la investigación, que al jugador tenga los créditos necesarios para mejorar la investigación, y que la investigación no esté en su máximo nivel. En caso contrario, se entregará el error correspondiente.

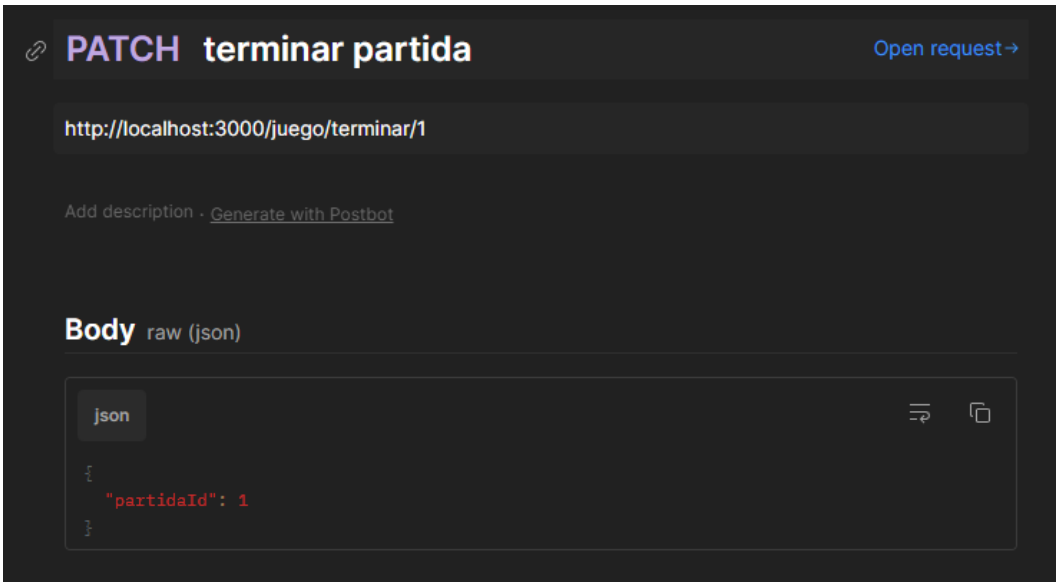
11. Cambiar crédito



En este endpoint lo que se hace es que se le cambia la cantidad de créditos a un jugador de acuerdo con la cantidad y al jugador indicado en el body.

Como todavía no es programada el *feature* de admin, todavia no es incorporado al front, por lo que solo puede ser testeado en el backend

12. Terminar partida

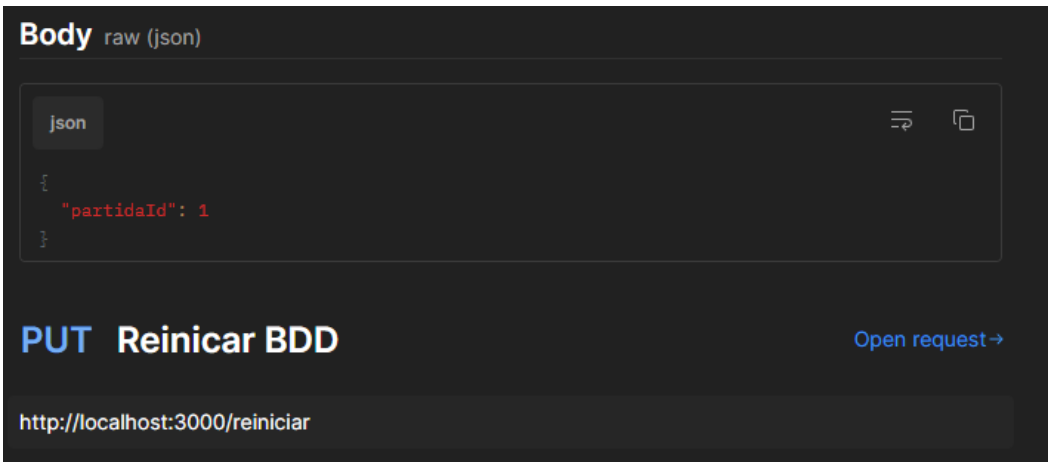


De la misma nera que en el endpoint anterior, todavía no es incorporado en el front porque no se cuenta con un rol de usuario, por lo que solo es testeable en Postman

En este endpoint, lo que se hace es forzosamente terminar la partida. Se entrega el id de la partida que se quiere terminar en el body. Primero verifica que esta partida exista, luego que haya sido comenzada ya, y finalmente que no haya sido

terminado previamente. En caso de que cumpla con todos, se buscan los jugadores que todavía estén activos, se da como ganador el jugador con más créditos, y se da por finalizada la partida.

13. Reiniciar BDD



Finalmente, este endpoint fue programado con el fin de facilitar el testeo vaciando las bases de datos.

4. Mockups Actualizados

Se planean mantener la mayoría de los mockups de la primera entrega, solo que no pudieron ser implementados al 100% por tiempo

Imagen 1: Landing Page. Fuente: Elaboración Propia

Imagen 2: Login/Registrarse. Fuente: Elaboración Propia

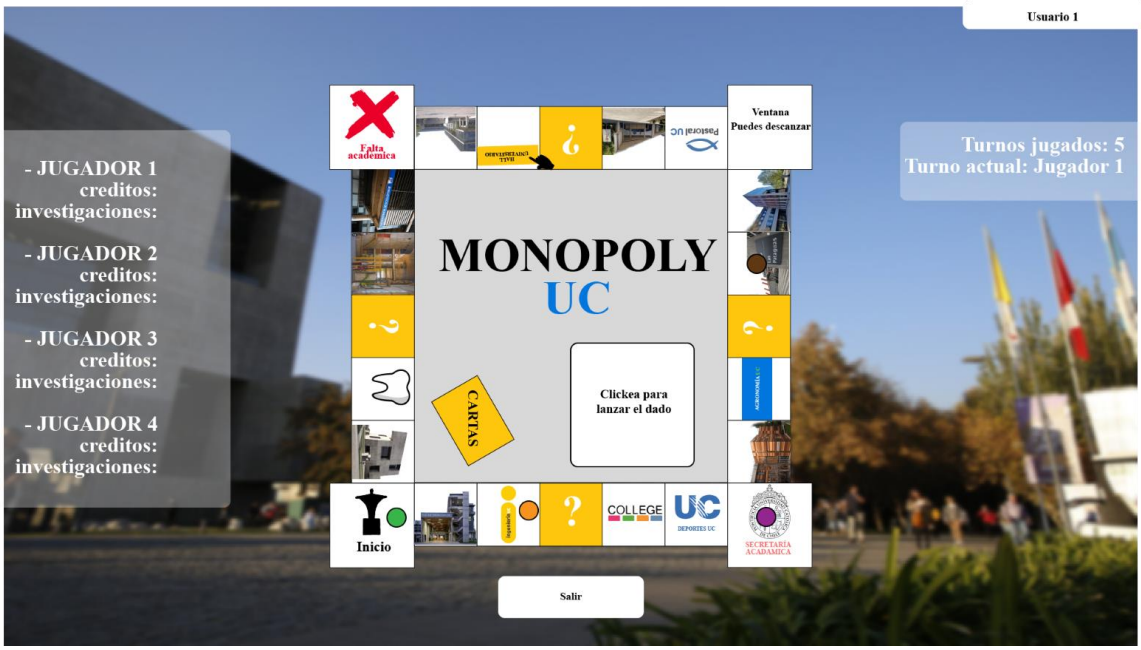


Imagen 3: Jugando. Fuente: Elaboración Propia

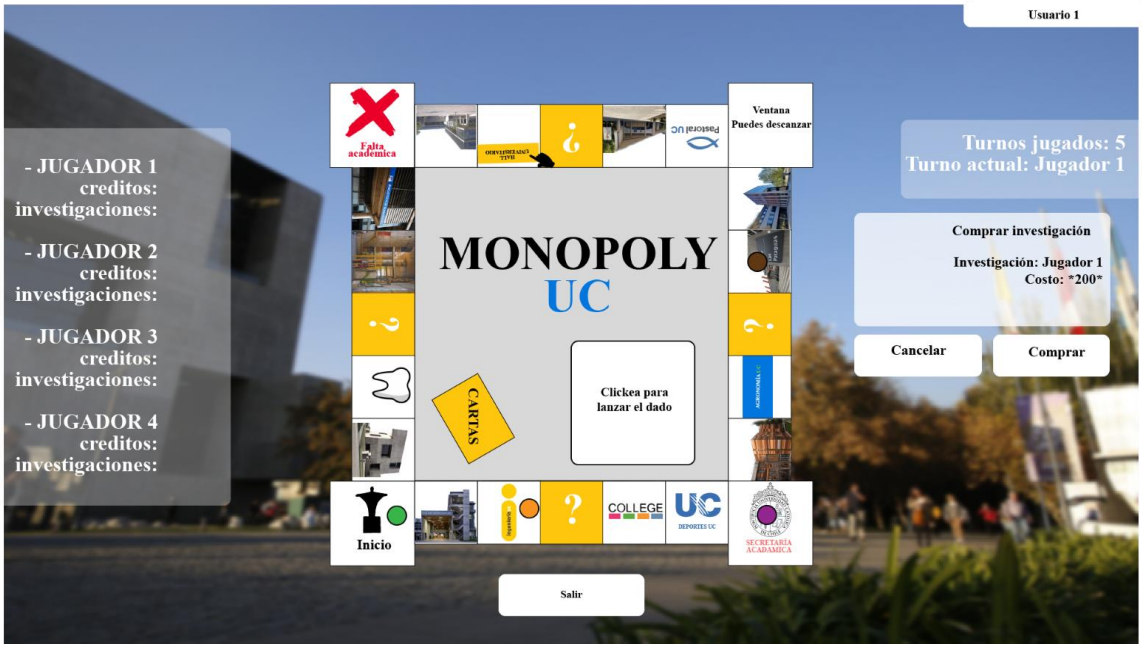


Imagen 4: Comprar una investigación nueva. Fuente: Elaboración Propia

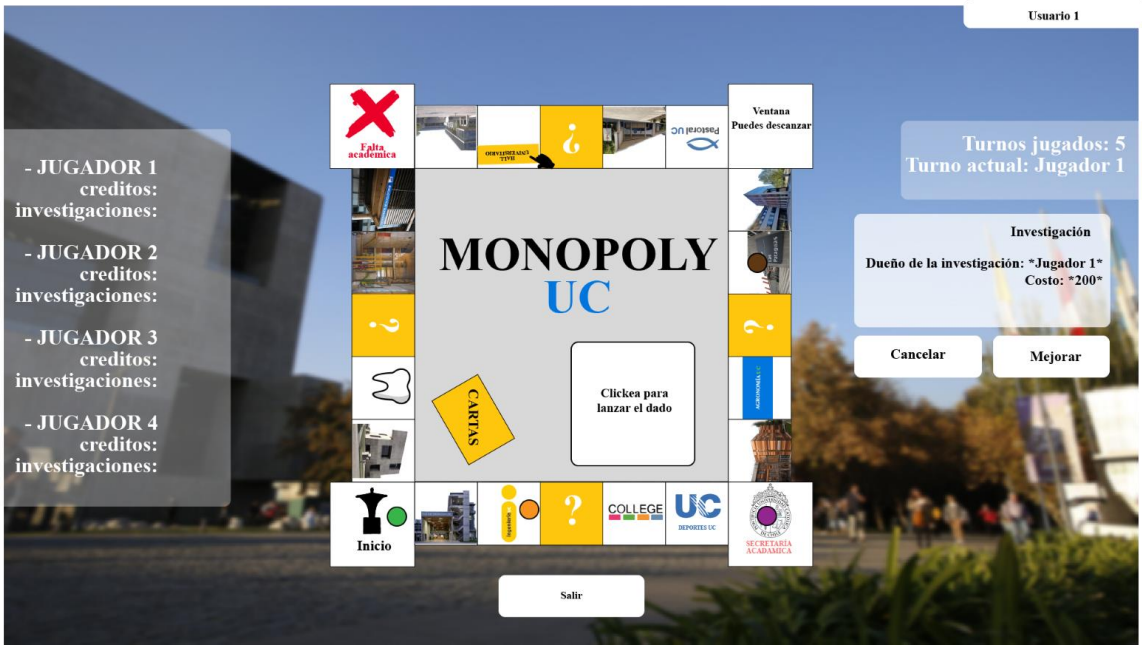


Imagen 5: Mejorar investigación. Fuente: Elaboración Propia

Imagen 6: Jugador eliminado. Fuente: Elaboración Propia

Imagen 7: Partida terminada. Fuente: Elaboración Propia

5. Consideraciones finales

Para esta entrega, lo que se realizo fue el desarrollo del frontend mediante el uso de React modular, y la integración de este con el backend.

Se realizó tanto una página de Index, donde se muestran elementos como el tablero, la lista de espera (no funcional todavía) y un botón que permite ir a la vista para iniciar sesión.

En la vista de iniciar sesión da la opcion de tanto iniciar sesión como registrarse. Ambas son 100% funcionales y están conectadas al backed. Para el manejo de sesión, se hizo uso del jsonwebtoken como se indicaba en las capsulas del curso, y toda la lógica de usuarios,

jugadores, etc, estam vinculadas al uso del token en el LocalStorage. El token era pasado de front a back a través de Headers, y en el back se recuperaba la información necesaria a partir de este, principalmente el id de usuario. A partir del usuarioId, podíamos acceder a sus jugadores, estados, etc. También cabe mencionar que se hizo uso del hasheo de contraseñas con bcrypt como se hizo en las capsulas.

Una vez iniciada la sesión, se devuelve al index, pero se aprovechó el uso de componentes y su opción de utilizarlos para ver que mostrar. Si el usuario se encuentra autenticado (iniciado sesión) se verifica a través del token y se le muestran unos botones diferentes, mostrando la opción de unirse a la partida y cerrar sesión.

Si el usuario aprieta la opción de unirse a partida (y la partida existe: detallado en el endpoint de crear partida), será redireccionado a /partida, donde nuevamente se muestra el tablero, pero se pueden ver la posición de los jugadores en el juego a través de fichas en las casillas. En este, se da la opción de tirar dado o comprar investigación de acuerdo sea el caso. También cuenta con un botón para refrescar la vista de la posición de los jugadores, ya que como no se cuenta con webSockets, si un jugador juega, los cambios se producen en la base de datos, pero no hay nada que gatille la actualización del componente en el otro jugador. Con todo lo antes mencionado, se cumple con los endpoints de sesión y de juego integrados. Mencionar que esta vista no fue desarrollada en mayor manera por temas de tiempo.

Finalmente, en el documento del backend, en la carpeta entregables/entrega3, se entrega la documentación con el detalle sobre como ejecutar el proyecto y testearlo, y también los mockups actualizados, que en verdad son los mismos, solo que no pudimos implementarlos al 100% por temas de tiempo