

Trabajo Social Comunitario 5

PROFESORA:

* Sabrina Abran

DATOS DE LOS ESTUDIANTES:

* Gastón Capretti
* Matias Azcui
* Martín Ambrueso

Contenido

[Estado de avance 2](#_Toc77412548)

[Guía de ingreso/utilización 3](#_Toc77412549)

[Proyección y planificacion 4](#_Toc77412550)

[Reflexión final 4](#_Toc77412551)

[Anexo técnico 6](#_Toc77412552)

[Despliegue local 6](#_Toc77412553)

[Configuraciones 6](#_Toc77412554)

[Swagger de documentación 8](#_Toc77412555)

[Endpoints 9](#_Toc77412556)

[Consideraciones para escalabilidad 9](#_Toc77412557)

[Consideraciones de seguridad 10](#_Toc77412558)

# Estado de avance

Desarrollar detalladamente los objetivos propuestos para el trabajo en el grupo y los motivos de las decisiones que fueron tomando en el proceso de investigación y desarrollo. Es importante que den cuenta de las diversas opciones que fueron probando y desechando en el camino incluyendo un listado de ventajas y desventajas de cada una de ellos. En este punto, sugerimos que recuperen lo escrito en el informe preliminar.

Indicar dudas, dificultades e interrogantes que fueron surgiendo en el trayecto de la cursada en relación con la propuesta de trabajo.

En este punto fue de vital importancia la comprensión del requerimiento. Debido a que comenzamos a trabajar en un proyecto que ya estaba comenzado, tuvimos que entender la parte funcional pero la parte técnica. Nos encontramos con que no todos los integrantes del equipo conocíamos las tecnologías utilizadas, y algunas las conocíamos con más o menos profundidad.

Los ítems que fuimos avanzando a lo largo de la cursada fueron:

* Se agregaron validaciones para el control de datos en los servicios que fueron desarrollados por los compañeros en las cursadas anteriores.
* Se desarrolló e implementó el servicio para la generación de usuarios finales como administradores. Vimos necesario avanzar con este servicio para allanar el camino a los futuros compañeros programadores. Encontramos esto de gran importancia para terminar de entender qué tipos de usuarios iban a trabajar con el sistema y dejar este requerimiento completamente definido.
* Se implementó un sistema de seguridad mediante la lógica correspondiente para la autenticación de la api (SignUp, SignIn). Esto es fundamental para la escalabilidad de la aplicación e integridad de la información.
* Se desarrolló el servicio que devuelve los datos del formulario para darse de alta en el sistema.

# Guía de ingreso/utilización

Realizar un breve instructivo o manual de procedimiento donde detallar el modo de ingreso y uso de lo producido hasta ahora.

Como guía de ingreso se debe realizar un GIT CLONE del siguiente repositorio para ver el código fuente de la API:

Para utilizar los servicios desarrollados de la API se generó el siguiente Swagger donde se visualiza los diferentes ENDPOINTS que se pueden consumir desde el FrontEnd. Este provee la información con la que debe realizarse el llamado (request) y qué información va a devolver este servicio (response):

# Proyección y planificación

Delinear una planificación tentativa de tareas sugeridas paso a paso para las/los compañeras/os que continúen su labor.

Si tuvieran que planificar lo pendiente, medir el tiempo de trabajo y los principales desafíos cuales serían.

Planificación:

1)Descargar el código fuente

2)Lectura del código para su entendimiento

3)Desarrollar el servicio bajo un endpoint que reciba la información con los datos que se llamaron en el formulario, guardarlos en una base de datos relacionados al usuario con el que se registró

4)Ponerse de acuerdo con el equipo de front end para decidir si ellos envían las coordenadas o desde el backend se recuperan por medio de una api. Este dato es fundamental para poderlo mostrar en un mapa.

5)Desarrollar el resto de los servicios llamados desde el menú del frontend.

# Reflexión final

Reflexionar en cuanto a las posibilidades, potencialidades o dificultades de vinculación de su disciplina con lo que denominamos economía popular. ¿Qué otros aportes imaginan para este sector?

Nosotros como estudiantes de la UNDAV entendemos que tenemos la responsabilidad de involucrarnos dentro del territorio para conocer las problemáticas en la cual la sociedad se ve afectada en la actualidad.

Si bien uno cuando arranca esta carrera se enfoca en aprender y desarrollarse desde una perspectiva individual para cumplir sus objetivos personales, las materias de TSC nos muestran que el desarrollo individual sin involucrarse socialmente no es suficiente si uno no se compromete o al menos no entiende la realidad social en la que vivimos. De esta manera mediante la construcción del conocimiento que fuimos adquiriendo a lo largo de estos años en contacto estrecho con el territorio hemos entendido lo complejo que es vivir en una economía popular, donde en nuestros proyectos de la facultad nos ha tocado trabajar con fábricas recuperadas, cooperativas, radios populares, comedores. De esta manera hemos entendido lo difícil que es subsistir ante el avance de la globalización y capitalismo salvaje en el que vivimos.

Además, como futuros ingenieros en informática tenemos la posibilidad de ante una problemática en particular que se nos presente tener la posibilidad de modelar esta situación para luego poder resolverla ya sea realizando un desarrollo, trabajando en conjunto con otros compañeros y equipos.

Es por esto que la combinación en involucrarnos en este proyecto como así relacionarlo con nuestra profesión fue algo muy grato para todos los integrantes de nuestro equipo, en donde lo hemos disfrutado y pudimos aportar nuestro trabajo para este proyecto.

# Anexo técnico

En esta sección se plantean todos los detalles técnicos, como así también los pasos de instalación y detalles a tener en cuenta para el escalado horizontal y vertical a futuro. Plantearemos los cuestionantes que tuvimos a lo largo de este trayecto y las decisiones por las cuales optamos y los por qué.

## Despliegue local

El proyecto en su totalidad es JavaScript (NodeJS), con lo cual, previamente se debe descargar NodeJS, que también cuenta con los gestores de paquete, en este caso npm, enlaces:

<https://nodejs.org/en/>

Cuenta con paquetes para todos los ambientes y sistemas operativos.

También se configuro un Dockerfile para levantar el ambiente de manera mas sencilla si se prefiere, previamente, también, deberá descargarse Docker:

<https://docs.docker.com/get-docker/>

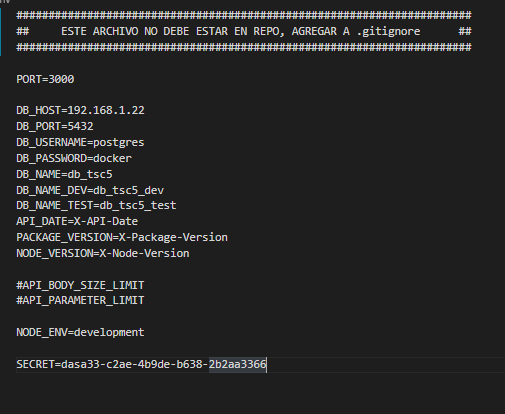
La base de datos esta en postgres sql, con lo cual también se deberá configurar un servidor para ello, puede descargarse pgadmin, y cuenta con toda la suite, o correr una imagen docker parametrizada:

pgAdmin: <https://www.pgadmin.org/>

Docker: <https://hub.docker.com/r/dpage/pgadmin4/>

## Configuraciones

A la hora de desplegar hay ciertas cosas a tener en cuenta, el proyecto cuenta con un archivo de variables de entorno, nombrado como .env. En el mismo están todas las configuraciones por ambiente, en este caso tener en cuenta que vamos a trabajar siempre sobre development, pero esta preparado para un circuito CICD para futuras integraciones.



Es importante aclarar que este archivo jamás debe estar en repositorio, en este caso se hizo ya que no esta productivo.

Como se puede ver, cuenta con configuraciones de todo tipo, pero tener en cuenta que tiene una secret y credenciales, es recomendable que, a futuro, al salir a producción, se configure un Vault de credenciales para ser inyectadas.

Prestar principal atención a las variables que comienzan con DB\_, ESTAS SON LAS PRINCIPALES QUE SE DEBEN CAMBIAR EN FUNCION DE LA CADENA DE CONEXIÓN DE LA BASE DE DATOS QUE TIENEN LEVANTADA LOCALMENTE.

Server={192.168.1.22};Database={db\_tsc5};Port={5432};UID={postgres};Password={docker}

El servidor de backend Node se va a valer de todo esto para trabajar, aportando estas configuraciones no va a ser necesario modificar mas nada, solo correr los comandos siguientes:

**# instala todas las dependencias necesarias, las lee del package.json**

npm install

**# crea la base de datos db\_tsc5**

npx sequelize db:create

**# corre la migracion de modelos de datos**

npm run migrations

**# instala el swagger de configuración**

npm run swagger-autogen

**# corre el servidor Node**

npm start

O también puede correrlo con Docker:

# descarga la imagen de docker y corre todos los stages del Dockerfile

docker build . -t tsc\_backend

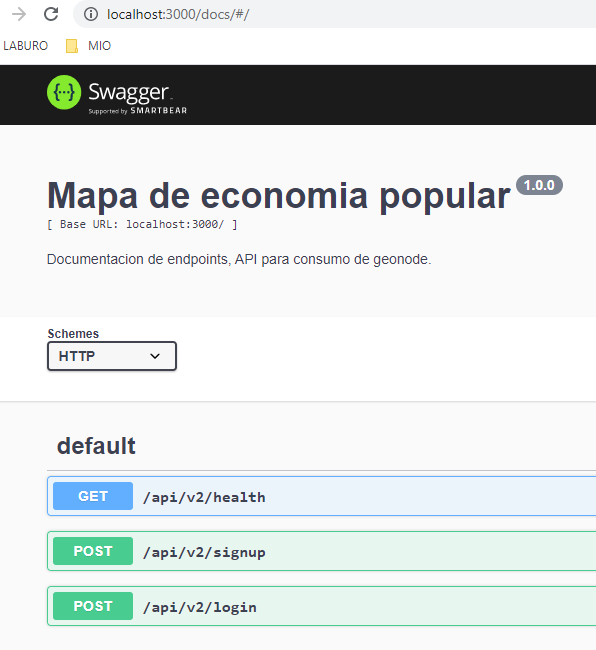
# corre el contenedor anteriormente creado en puerto 3000

docker run -p 3000:3000 -d tsc\_backend

## Swagger de documentación

El proyecto cuenta con swagger para obtener mas información de los endpoints de la API, de esta manera se pueden hacer pruebas preparadas para conocer un poco mejor y tomar confianza con los recursos expuestos.

Este documento se expone en <http://localhost:3000/docs/> una vez corriendo el servidor.



## Endpoints

BASE URL: <http://localhost:3000/>

'/api/v2/business-area'

'/api/v2/categories'

'/api/v2/categories/:id/subcategories'

'/api/v2/entities'

'/api/v2/signup'

'/api/v2/login'

Para mejor apreciación, observar con Swagger docs

## Consideraciones para escalabilidad

El proyecto intentamos hacerlo lo mas reutilizable y desacoplado posible, siempre esta abierto a mejoras, pero vamos a indicar cuales son los archivos a editar en el eventual caso de querer agregar servicios:

En caso de querer agregar mas modelos de datos a la base de datos, tener en cuenta que se esta usando un ORM (Object Relational Mapping), con lo cual, no se están ejecutando querys sql desde el código, sino que se abstrae esa lógica al framework de Sequelize ORM. Va a ser necesario que cree un modelo:

npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string

Esto nos va a crear una nueva migración, y debemos correrla para que impacte:

npx sequelize-cli db:migrate:undo

Esto va a crear todos los archivos y tablas necesarias en postgres para poder mapear objetos desde el backend.

En caso de agregar más endpoints, se deberá colocar lógica en archivos independientes en las siguientes carpetas:

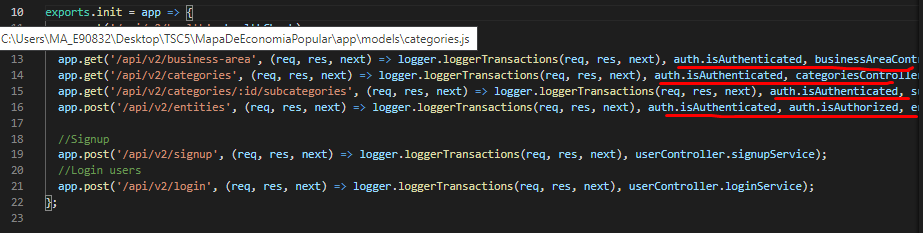
App/controller -> capa de validaciones, llama al servicio para lógica de datos

App/Services -> lógica de datos, íntima relación con ORM

App/routes.js -> se agregan las rutas con middlewares

## Consideraciones de seguridad

Se construyo toda una capa de servicio para autenticación, actúa como middleware en cada endpoint a ser ejecutado. De manera que valida previamente y permite o deniega.



También se diseñaron los endpoints para autenticación y autorización, la implementación de seguridad esta corriendo con Json Web Token, pero se plantea como posible mejora:

* El algoritmo de firma digital actual es HMAC + SHA256, es conveniente generar claves RSA y configurarlo con algoritmo de firma asimétrica RSASSA-PKCS1-v1\_5 + SHA256, actualmente la verificación es simétrica y segura, pero esto le daría valor agregado.
* Hacer una gestión mas compleja de los tokens, actualmente tienen tiempo de caducidad fijos, no hay lógica para perfiles
* La autorización por admin esta basada en un campo booleano en base de datos de users, es necesario armar un modelo de usuario, roles y permisos, la lógica de backend llevaría bajo impacto, es solo diseñar los modelos de base de datos.
* El logging de auditoria se esta guardando en un archivo Audit.log, se agrego en el middleware de logger un objeto de configuración para Logstash, esta funcionando, pero se deberá desplegar instancias ELK en el ecosistema.
* La encripcion de credenciales en la base de datos de usuario no tiene lógica de semilla, por cuestiones de tiempo se opto por encriptar con AES pero es altamente recomendable agregar lógica de semilla, lamentablemente no pudimos llegar.
* Credenciales de ambiente, recomendable colocar un Vault de credenciales en el ecosistema productivo.
* Hacer una gestión mas compleja de los payloads de JWT, si bien cumple, se pueden agregar muchas más validaciones.
* Adicionarle al servidor productivo TLS.
* Hacer una gestión mas compleja de los headers http.