UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES


| **Integrantes** |
| --- |
| Lobo, Brian |
| Taberna, Alexis |
| Bulbulian, Juan Pablo |

**Ayudante Asignado:** Perez Albiero, Alejandro


**Materia:** (TACS) Tecnologías avanzadas para la construcción de software

# *Índice*

# *Toma de decisiones*

En este documento se habla de la toma de decisiones relacionadas con tecnologías y de negocio que se acordaron para el desarrollo de este trabajo práctico.

## Requerimientos funcionales y no funcionales a más detalle:

- https://docs.google.com/document/d/1Dvydn8WtLzCuI7VqrAdAHL8FNXPYnDTZrio7c-WV_Kk/pub

## Requerimientos pedidos por el TP:

- **Sistema de versionado:** Github
- **Lenguaje:** java, se decidió usar la versión 1.8 por ser la más estable y nueva.
- **Manejo de dependencias:** Maven
- **Lugar donde corre la aplicacion:** tecnologías Cloud
- **Metodología:** ágil
- **Tipo de Aplicación:** Web

## Tecnologías Elegidas:

- Front-end
  - **Angular 5:** Se utilizó esta versión por ser la más reciente y conocida por el equipo, otras opciones fueron AngularJs, React y JavaScript puro, pero por las ventajas de sus herramientas y conocimientos básicos de los miembros del equipo (además de la inmensa comunidad que trabaja y da soporte detrás) se inclinó por esta opción.

  - **NodeJs:** Aunque posea muchas más herramientas se optó por usar su gestor de dependencias **npm** y su facilidad de uso a nivel programación e instalación, esto también posibilitó tener el front separado del back, agregando más portabilidad al producto.

  - **Bootstrap 4:** Se utilizaron templates pre-hechos, lo que favoreció en el periodo de desarrollo visual y permitio administrar los tiempos para un fortalecimiento de la seguridad y la integración con el backend de la aplicación, ya que se consideró que el *look-and-feel* es un aspecto importante, pero lo es más la seguridad que se debe ofrecer al trabajar directamente con dinero virtual. Como agregado el template utilizado posee la naturaleza de ser responsive, brindándole de fácil aprendizaje y una experiencia similar al usarlo tanto desde un smartphone como de una PC.

- Telegram Bot
  - **Descripción:** Se trata de una aplicación que utiliza una biblioteca recomendada por la compañía Telegram para interactuar con la API que ellos mismos ofrecen al público. El bot puede ser encontrado en el directorio de Telegram bajo el nombre ***TACSCryptoBot***. Consta de cinco comandos, los cuales permiten cumplimentar las user stories encomendadas en el enunciado del Trabajo Práctico.

  - **Tipo:** Se trata de un ***long polling bot***, a diferencia del tipo ***webhook***, la aplicación monitorea los servidores de Telegram en busca de updates que se hayan dado en el bot. La decisión se tomó en base a la posibilidad que daba el servicio de Cloud en Heroku, si se elegía el tipo webhook, se requería tener asignado un puerto fijo, cosa que Heroku no brinda en las cuentas gratuitas.

  - **Implementación:** Es muy sencilla, desde una clase main se invoca a la clase *TACSCryptoBot.java*, la cual tiene la especificación de los comandos, los cuales están dentro de el paquete command. Todos heredan de una clase común *Command.java* y corren su metodo *exec*.

  - **Seguridad:** El Token del bot, el cual fue provisto por la compañía Telegram al crear el bot es pasado por parámetro de línea de comando al iniciar el bot. De esta manera, solo es conocido por el creador del Bot, que es quien tiene la responsabilidad también de correrlo en el entorno Cloud.

  - **Deploy:** La aplicación se encuentra actualmente corriendo en el servicio de Cloud Heroku.

  - **Tecnologías:** La aplicación está desarrollada en **Java 1.8**, la decisión se tomó al investigar que las bibliotecas y frameworks disponibles en la comunidad harían que la implementación sea bastante sencilla.
    Se utilizó el cliente web de **Spring Framework** para las request al Backend, y la biblioteca **Jackson** para el manejo de los datos en formato **JSON**.

- Backend
  - **Estructura:** Se optó en una estructura de mono-aplicación la cual era más simple de crear y mantener para lo que el negocio requería. Al no poseer una lógica demasiado compleja y una cantidad de LOC razonable, se falló a favor de un modelo clásico conformado por una estructura de proyectos en módulos relacionados por un POM padre para su sencillo despliegue. El proyecto cuenta con un módulo Core, donde se encuentran las dependencias y clases transversales a toda la aplicación, un model donde se encuentra los objetos de negocio (POJO's), un proyecto Dao para la capa de acceso de datos,un proyecto service para la lógica de negocio, uno rest para el manejo

de los pedidos http y por último un proyecto apiWeb con todo lo necesario para levantar la aplicación web.

- **Lenguaje:** Se seleccionó Java 1.8, la versión 1.7 por su mejoría en sus herramientas para el manejo de listas, conceptos funcionales e interfaces.

- **ORM:** Se escogió Spring Data como framework orm por su robustez, amplia documentación y facilidad a la hora de persistir objetos.

- **Web-Service:** Se optó por una composición entre Spring y Jersey.

- **Server:** La aplicación corre sobre un servidor Apache Tomcat, dado a su facilidad de despliegue tanto en un entorno local como en uno cloud, su facilidad de uso y su numerosa documentación (además de la considerable comunidad que le da uso).

- **Programas para desarrollo:** Tales como Eclipse, Dbeaver, Insomnia, Postman, Visual Studio Code,Sublime Text, Git Kraken, entre otros, son herramientas para desarrollo Free u Open Source, además de multiplataforma, para su disponibilidad y consistencia según el SO de cada integrante.

- **Base de Datos:** La persistencia se realiza en una base de datos no relacional, más específicamente MongoDB, en un principio se tuvo en consideración una base sql, pero se concluyó que es dispensable la estructura de relación entre datos que estas poseen (es cierto que estas relaciones existen, pero no son de tal magnitud como para decantar en este tipo de base de datos). MongoDB es una base nosql basada en archivos simple de instalar, configurar y muy versátil, siendo una herramienta sencilla para programar y ágil a la hora de obtener datos de esta; Es cierto que parte de la lógica que se modeló en stored procedures en estados iniciales del proyecto quedó asignada en el backend, pero a cambio se ganó algo muy simple y rápido para obtener objetos por ids, así como facilidad para almacenarlos.

## Administración del proyecto:

- **Herramientas:** Se empleó un sistema de tickets administrado por Trello de licencia Free  y de simple uso ya que es online y web.

- **Metodología**: El proyecto se desarrollo utilizando la técnica de Timebox Development, la cual deja fijo el calendario (Fechas de entrega) y prioriza funcionalidades a cumplir en cada entregable (Funcionalidades obligatorias y agregados), manteniendo la calidad de todos los entregables y por supuesto del producto final.

## Negocio:

- **Compra y venta:** Se decidió ir por comprar monedas a la aplicación, así como un “Banco”, no entre usuarios

- **Monedas Físicas:** Solo se trabaja con Dólar Estadounidense.

- **Depositos**: Seran aceptados por un usuario administrador desde una UI del sistema.

## Tipo de Aplicación

- Como la aplicación es WEB se decidió hacerla responsive, para que pueda ser utilizada en varios dispositivos sin importar SO o hardware.

- Se permitio que usuarios puedan manejar múltiples sesiones desde sus cualquier dispositivo, permitiendo una mejor UX.

- La aplicación fue pensada para agregar clientes que la consuman, siendo diseñada de tal forma que el dia de mañana se puedan utilizar sus servicios de una app nativa, de esta forma funciona Telegram, quien consume de los mismo servicios que la aplicación WEB sin distinción.

- También fue pensada para escalar, permitiendo tener instancias en varios servidores que, correctamente configurados, puedan trabajar en conjunto y así escalar horizontalmente.