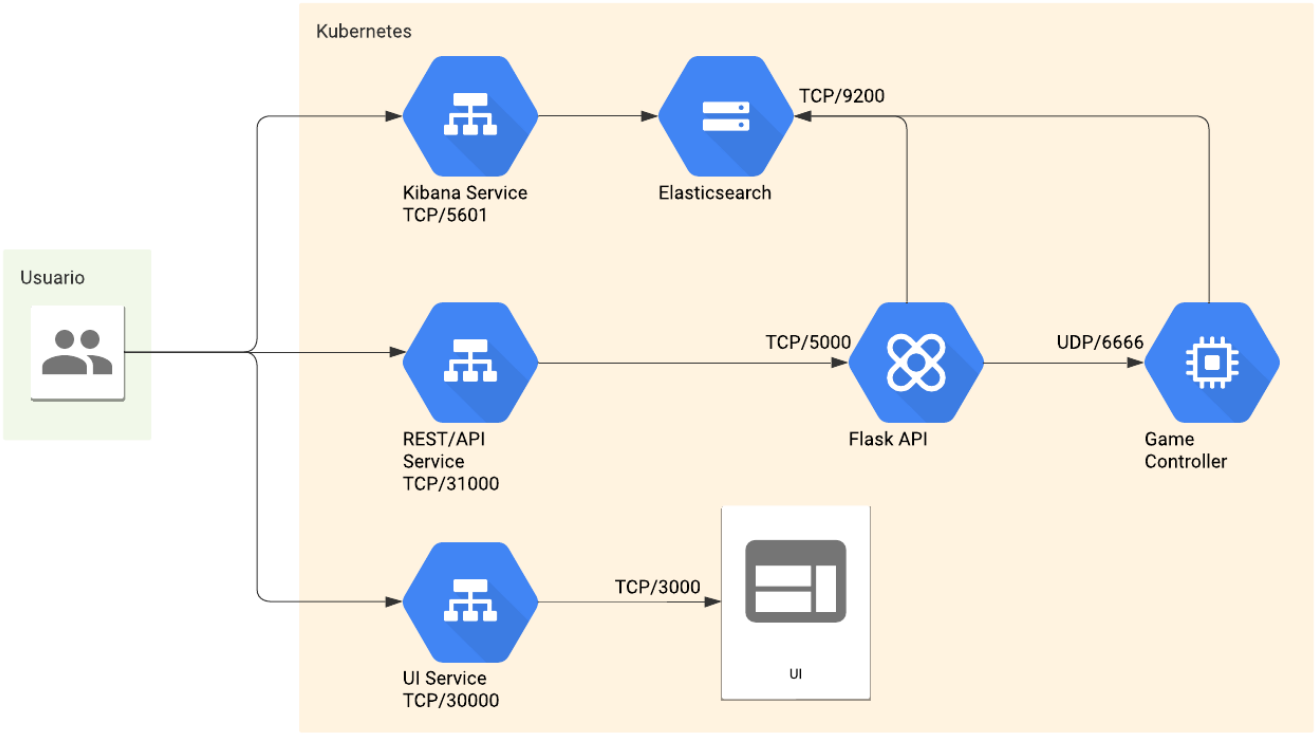
Documentación - Crazy Chess

Estudiantes:

- Ary-El Durán Ballestero | 2018102445
- Isaac David Ortega Arguedas | 2018189196
- Zhong Jie Liu Guo | 2018319114

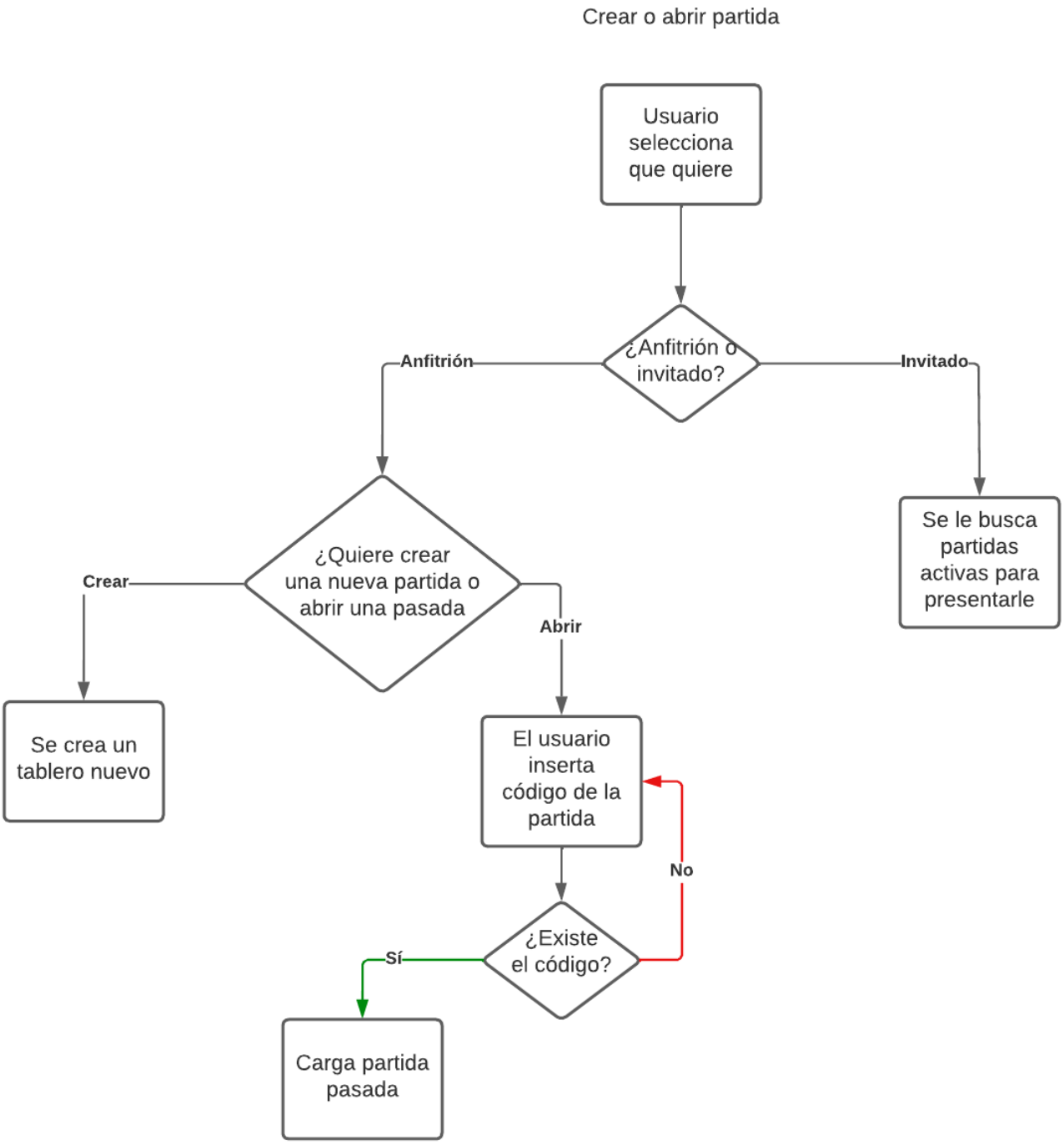
Diagrama de Arquitectura

La base de este programa se tomó de la especificación del proyecto. Es bastante similar en el sentido de los puertos y las conexiones.

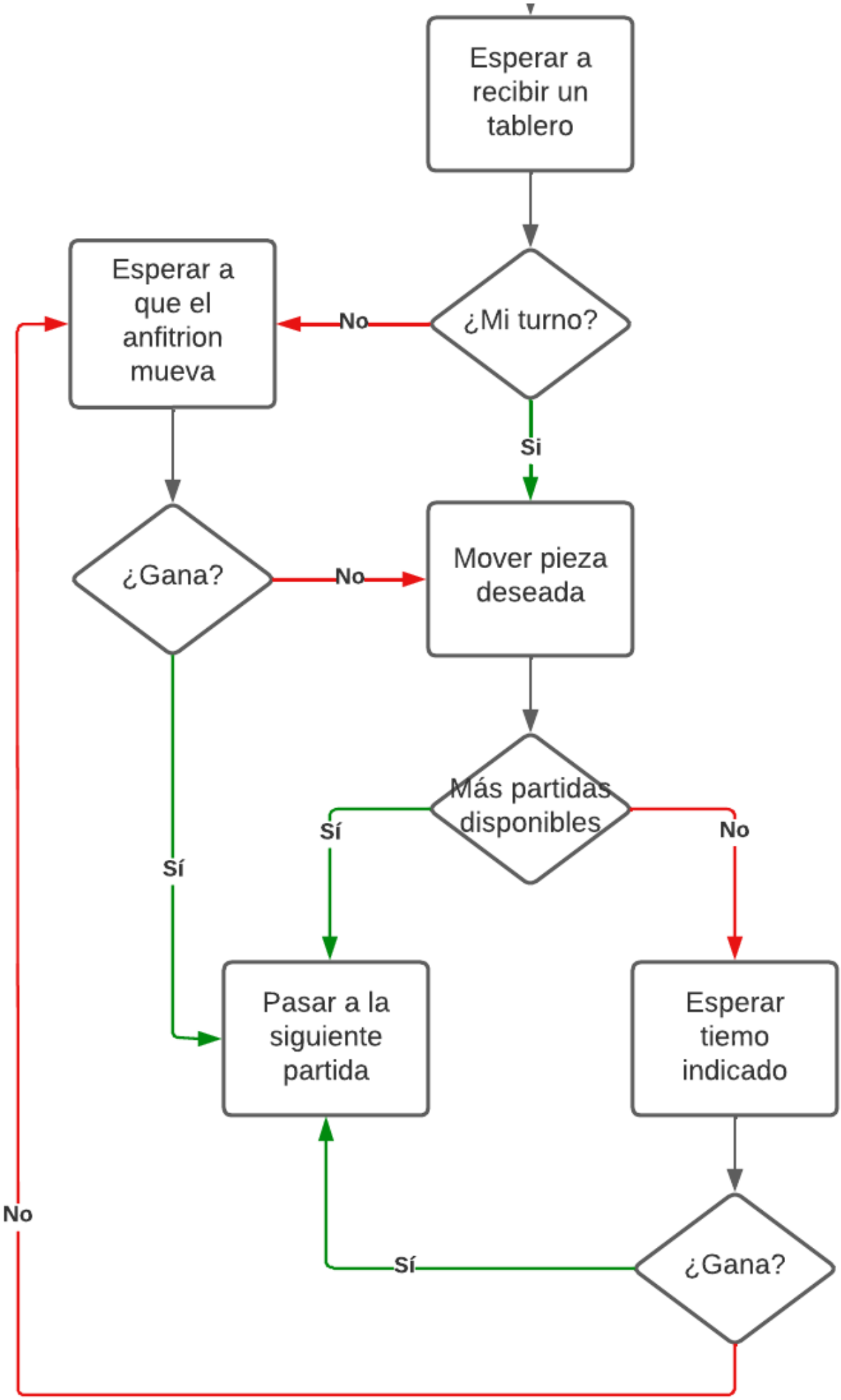


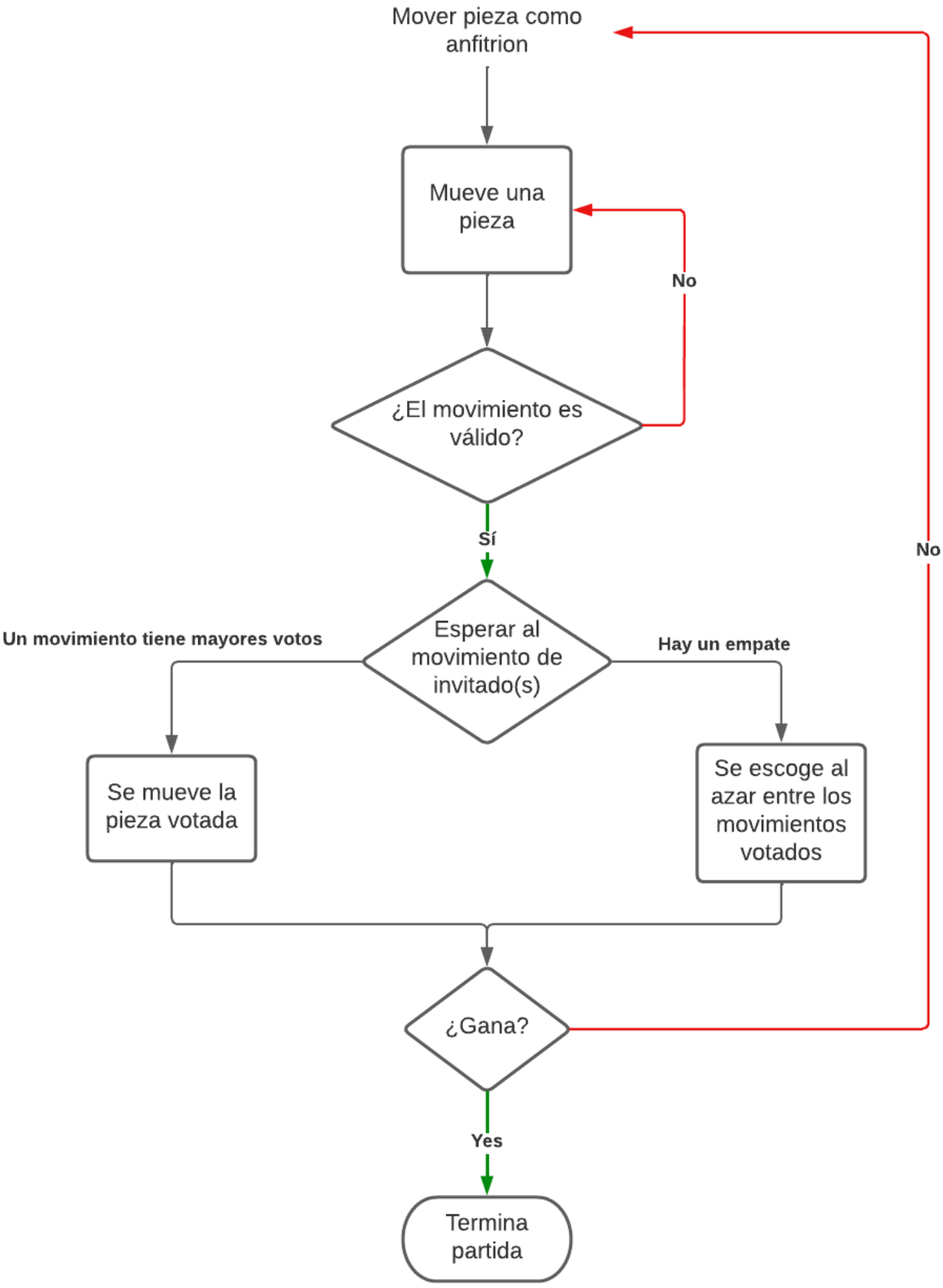
Diagramas de Flujo

Para este apartado, se mostrarán tres diagramas para las tres historias de usuario: crear un juego, jugar como invitado y jugar como anfitrión.


Crear o abrir partida
Usuario selecciona que quiere
¿Anfitrión o invitado?
Anfitrión
Invitado
Se le busca partidas activas para presentarle
¿Quiere crear una nueva partida o abrir una pasada
Crear
Abrir
Se crea un tablero nuevo
El usuario inserta código de la partida
No
¿Existe el código?
Sí
Carga partida pasada


Jugar como invitado
Seleccionar invitado


Esperar a recibir un tablero
¿Mi turno?
No
Esperar a que el anfitrion mueva
Si
¿Gana?
No
Mover pieza deseada
Más partidas disponibles
Sí
No
Sí
Pasar a la siguiente partida
Esperar tiemo indicado
¿Gana?
Sí
No



Prerequisitos para el proyecto

Para este proyecto, se asume que tiene instalado los siguientes programas:

- Docker Desktop(ver link)
- Kubernetes (Habilitar Kubernetes desde Docker Desktop)
- Helm (ver instalación)
- kubectl (Si no se instala por defecto, vea link)
- (Opcional) Lens (ver link)

Scripts para comenzar a usarlo

Los comandos se pueden dividir en dos partes: la creación de las imágenes y la corrida en Kubernete. El primero usará el docker-compose que ayudará a construir las imágenes necesarios de las aplicaciones. Con el programa Lens, puede revisar de forma visual el progreso de la instalación.

Instalación de Elasticsearch y Kibana

Se va a instalar de primero Elasticsearch y Kibana ya que se ocupan las credenciales de acceso para su uso. Se usará como base la guía Quickstart de Elastic Cloud en Kubernetes, pero se van a mostrar los comandos necesarios. Para una explicación más detallada, vea el link anterior. Los siguientes comandos son para una primera instalación.

```
kubectl create -f https://download.elastic.co/downloads/eck/2.4.0/crds.yaml
kubectl apply -f https://download.elastic.co/downloads/eck/2.4.0/operator.yaml
```

Luego, se usarán los archivos ubicados en la carpeta **db**, elastic.yaml y kibana.yaml. En una terminal, ubíquese en la carpeta **db**, luego haga pull a las imágenes si se requieren.

```
docker pull docker.elastic.co/elasticsearch/elasticsearch:8.4.0
docker pull docker.elastic.co/kibana/kibana:8.4.0
```

Cuando las descargas terminen, corra los siguientes comandos y espere un tiempo para que los servidores terminen de levantarse.

```
kubectl apply -f elastic.yaml
kubectl apply -f kibana.yaml
```

Para revisar si ha terminado, ejecute `kubectl get elastic` y `kubectl get kibana`. Si todo sale bien, aparecerá la salud (HEALTH) en verde.

Para acceder a kibana, se debe hacer port-forwarding con `kubectl` o hacerlo por medio de Lens.

```
kubectl port-forward service/quickstart-kb-http 5601
```

Luego, va a aparecer un log-in, el usuario es **elastic** y la contraseña puede averiguarse corriendo el siguiente comando: `kubectl get secret quickstart-es-elastic-user -o=jsonpath='{.data.elastic}' | base64 --decode; echo`. Si está en Windows, se recomienda correrlo en una terminal de WSL o de Unix.

Si quiere detener los pods, vaya a la carpeta **db** y corra los siguientes comandos.

```
kubectl delete -f elastic.yaml
kubectl delete -f kibana.yaml
```

Creación de imágenes

Ahora, hay que crear las imágenes para que se ejecuten en los pods. La ruta de la terminal tiene que ser en la carpeta **ProyectoOpcional**. Luego, se construirán utilizando el comando `docker-compose build <nombre-imagen>`. Va a tomar algunos minutos para descargar las dependencias.

```
cd <a la ruta del proyectoOpcional>
docker-compose build api
docker-compose build frontend
docker-compose build game-controller
```

Si necesita eliminar las imágenes creadas y todo lo relacionado a ello se usa `docker-compose down --rmi all`. Si hizo un cambio en el código, simplemente ingrese `docker-compose build <nombre-imagen>` de la imagen modificada.

Levantar los pods en Kubernetes

Ahora, se van a levantar los pods en Kubernetes con las imágenes apropiadas utilizando helm charts. Primero hay que dirigirse a la carpeta **app-deployment**. Luego se utiliza el siguiente comando para montar la arquitectura en el Kubernete.

```
## Crear los pods
helm install app-deployment .
## Detener y eliminar los pods
helm delete app-deployment
```

Lo anterior va a levantar múltiples pods y servicios relacionados al proyecto. Espere unos segundos y para utilizar CrazyChess, abra su navegador y escriba `http://localhost:30000/` para abrir la UI. Si desea ver la API, la url es la siguiente: `http://localhost:31000/` Si todo salió bien, le aparecerá el menú principal de la aplicación.

Recomendaciones

1. Hacer la integración de kubernetes con las aplicaciones en una etapa temprana para evitar contratiempos por conexiones fallidas.
2. Medir mejor el tiempo entre etapas para mejorar la etapa de desarrollo del proyecto.
3. Investigar más las funciones de Kubernetes en cuestiones de redes para conocer lo necesario para las conexiones entre pods para que en el futuro no haya atrasos.
4. La etapa de investigación de herramientas debe ser de menor tiempo a lo planeado e incluir código inicial para probar lo básico de esas herramientas.
5. Utilizar con más frecuencia las ayudas externas (como el profesor) para despejar las dudas en un menor tiempo que tratando de buscarlo en internet por varios días.
6. Apuntar a una mejor fecha para tener un producto preliminar para luego hacer las respectivas pruebas unitarias ya que, por el tiempo, no se pudieron enfocar en este aspecto.
7. Coordinar mejor el tiempo entre las asignaciones de otras materias ya que hubieron comflictos en la elaboración de avances para este proyecto.
8. Separar el trabajo por funcionalidad ya que hubieron muchas pausas para esperar que un compañero terminara una parte específica.
9. Trabajar con la arquitectura lo más parecida posible a la que van a correr los pods para evitar problemas durante deployment.
10. Usar mejor la seguridad que provee Elasticsearch.

Conclusiones

1. Se utilizaron las herramientas de Docker, Helm y Kubernetes para lograr la automatización de varias aplicaciones al levantar pods con varios programas, entre ellos react, flask api, c y elasticsearch.
2. Se aprendió que Docker es una herramienta muy útil para estandarizar el despliegue de un programa completo ya que este permite correrlo en cualquier computadora que tenga los programas necesarios para usar Docker.
3. Flask es una herramienta que hace la creación de una RESTful API increiblemente sencilla.
4. Los sockets nos permiten la comunicación de varios programas que estan incluso en diferentes lenguajes sencilla.
5. Libcurl nos permitira usar restful apis que no tienen alguna biblioteca para armar las conexiones, hacerlas de manera manual.
6. Se observó la facilidad de levantar una estructura completa de software por medio de pods ya que estos son reemplazables y fácilmente instalados por medio de kubernetes y helms.
7. Se vio que los Secrets de Kubernetes facilitaron el manejo de credenciales personales como las de ElasticSearch sin tener que revelarlos de forma obvia.
8. Los servicios que se usan para comunicar varias réplicas de pods son servibles para coordinar las cargas de cada aplicación y asegurar su soporte a fallas.
9. Se lograron levantar diferentes pods con sus propios servicios para que puedan ser levantados en una máquina local y funcionar como si se hubiera instalado en la propia máquina.
10. Lastimosamente, no se logró completar la funcionalidad completa ya que faltan las interacciones entre el game controller y la base de datos por medio de curl debido a que no se logró implementar su uso.