

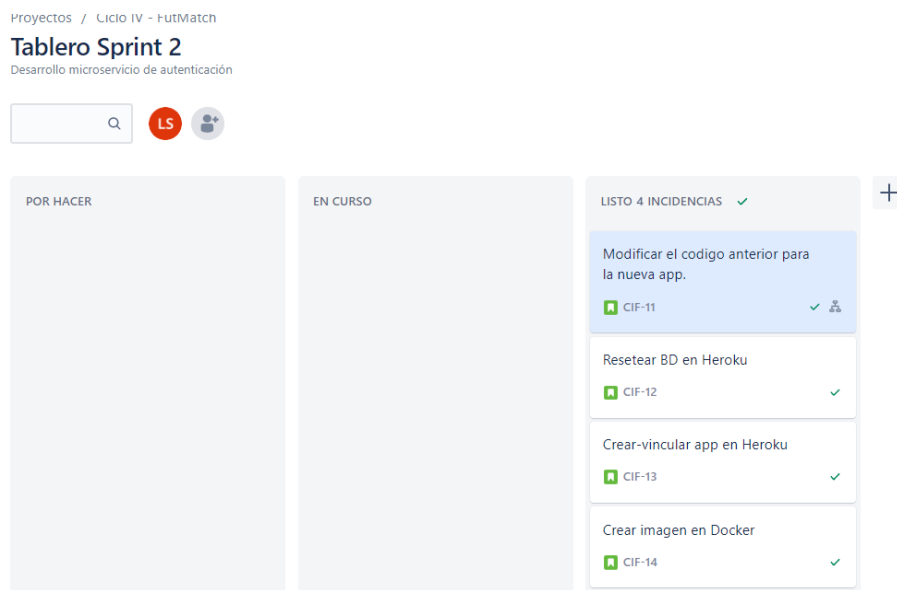
FUTMATCH

Informe Sprint #2

Jira

Para el sprint #2 se crearon las siguientes incidencias en la plataforma de Jira:

- Modificar el código anterior para la nueva app
- Resetear la BD en Heroku
- Crear/vincular la app en Heroku
- Crear Imagen en Docker



Para la primera incidencia se tenían las siguientes subtarefas:

- Actualizar modelos
- Actualizar vistas
- Quitar CORS y relacionados

La idea principal para esta incidencia es la actualización del código que se tenía del ciclo anterior, en dónde se deja solamente el modelo para usuario. Este código será mostrado más adelante en este informe.

Añadir epic / CIF-11

Finalizada ✓ Listo

Modificar el codigo anterior para la nueva app.

Adjuntar Añadir una incidencia secundaria Vincular incidencia

Descripción
El microservicio a modificar será el de autenticación

Incidentes secundarios 100 % hecho

| | | |
|--------|----------------------------|------------|
| CIF-15 | Actualizar modelos | FINALIZADA |
| CIF-16 | Actualizar vistas | FINALIZADA |
| CIF-17 | Quitar CORS y relacionados | FINALIZADA |

Detalles

Responsable Sin asignar

Etiquetas Ninguno

Sprint Tablero Sprint 2

Story point estimate Ninguno

Desarrollo Crear rama

Informador LS Lina Soler

Creado hace 6 días

Actualizado hace 7 minutos

Configurar

Además, con el desarrollo de este sprint se deja desarrollado el microservicio de autenticación que cumple con la historia de usuario #1 de nuestra app. Esta también fue actualizada en la plataforma de Jira.

Añadir epic / CIF-5

Historia de usuario #1

Adjuntar Añadir una incidencia secundaria Vincular incidencia

Descripción
"Como jugador de futbol quiero tener acceso a una aplicación donde pueda encontrar gente para organizar partidos de futbol en mi zona cercana."

Descripción: Desarrollar un login para que los usuarios se puedan registrar en la aplicación

Actividad

Mostrar: Todo Comentarios Historial Más recientes primero

LS

Añadir un comentario...

Consejo de expertos: pulsa M para comentar

Código (modelo usuario)

Para iniciar con la arquitectura de los microservicios, se realizaron las respectivas configuraciones en el archivo `requirements.txt` y en el archivo `futbolTinderProject/settings.py`

Se modificó la vista de Usuario *UserDetailView* teniendo en cuenta que ya no es necesario verificar que el token de la petición le pertenece al usuario cuya información se solicita.

```
userDetailView.py X
futmatch_authMS > futbolTinderApp > views > userDetailView.py > ...
1  from django.conf import settings
2  from rest_framework import generics, status
3  from rest_framework.response import Response
4  from rest_framework_simplejwt.backends import TokenBackend
5  from rest_framework.permissions import IsAuthenticated
6  from futbolTinderApp.models.user import User
7  from futbolTinderApp.serializers.userSerializer import UserSerializer
8
9  class UserDetailView(generics.RetrieveAPIView):
10     queryset = User.objects.all()
11     serializer_class = UserSerializer
12
```

Para esto, el componente API Gateway se encargará de verificar que el usuario se encuentra autenticado, por lo tanto creamos un nuevo archivo *TokenVerifyView* que incluye el método POST que se encargará de retornar el id del usuario en caso de que el token sea válido, haciendo uso de los métodos y serializers dispuestos por Simple JWT.

```
verifyTokenView.py X
futmatch_authMS > futbolTinderApp > views > verifyTokenView.py > ...
1  from django.conf import settings
2  from rest_framework import status
3  from rest_framework.response import Response
4  from rest_framework_simplejwt.views import TokenVerifyView
5  from rest_framework_simplejwt.backends import TokenBackend
6  from rest_framework_simplejwt.exceptions import InvalidToken, TokenError
7  from rest_framework_simplejwt.serializers import TokenVerifySerializer
8
9  class VerifyTokenView(TokenVerifyView):
10     def post(self, request, *args, **kwargs):
11         serializer = TokenVerifySerializer(data=request.data)
12         tokenBackend = TokenBackend(algorithm=settings.SIMPLE_JWT['ALGORITHM'])
13
14         try:
15             serializer.is_valid(raise_exception=True)
16             token_data = tokenBackend.decode(request.data['token'], verify=False)
17             serializer.validated_data['UserId'] = token_data['user_id']
18
19         except TokenError as e:
20             raise InvalidToken(e.args[0])
21
22         return Response(serializer.validated_data, status=status.HTTP_200_OK)
```

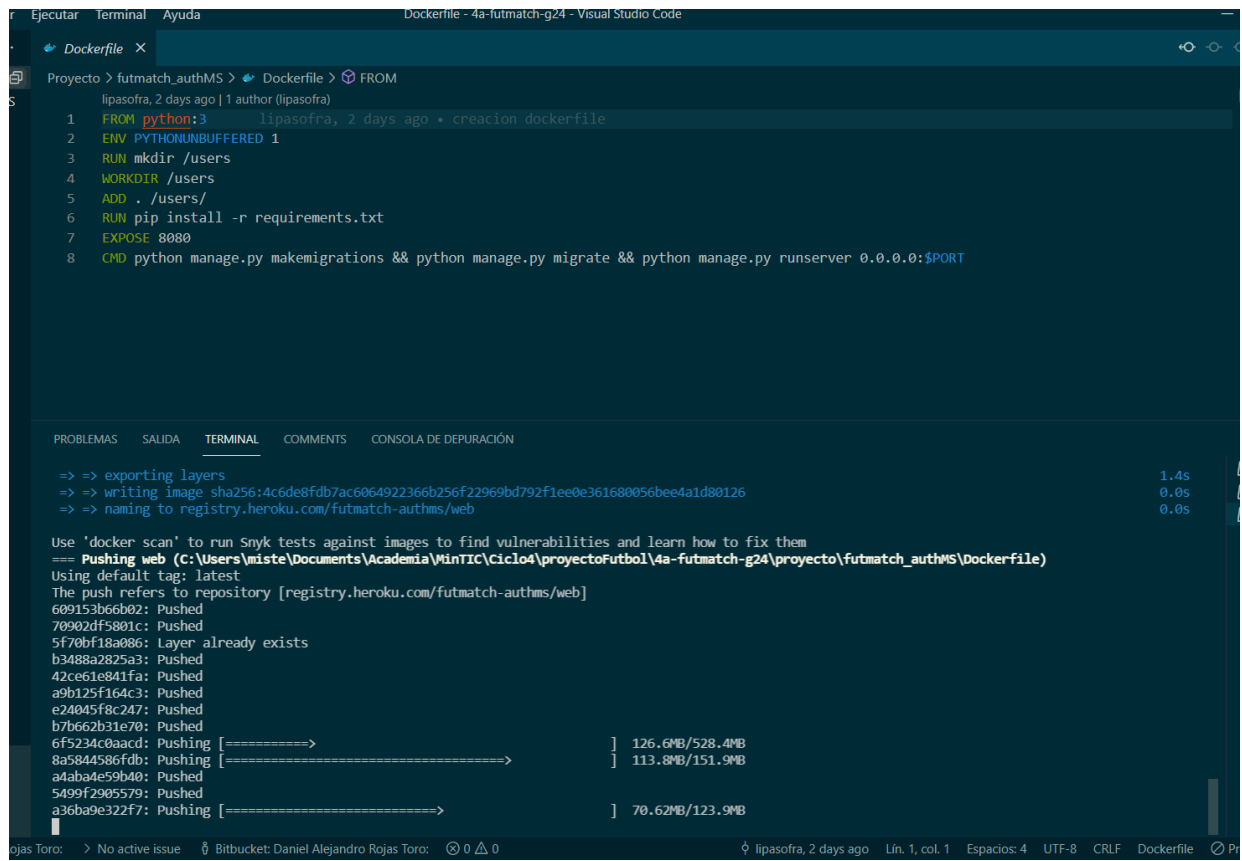
Asimismo, los archivos nuevos y modificaciones se registran como corresponden en el archivo `__init__.py` y, finalmente en lo que respecta al código, se asocia la nueva vista con una url del microservicio.

```
urls.py X
futmatch_authMS > futbolTinderProject > urls.py > ...
1  from django.urls import path
2  from rest_framework_simplejwt.views import (TokenObtainPairView, TokenRefreshView)
3  from futbolTinderApp import views
4
5
6  urlpatterns = [
7      path('login/', TokenObtainPairView.as_view()),
8      path('refresh/', TokenRefreshView.as_view()),
9      path('user/', views.UserCreateView.as_view()),
10     path('user/<int:pk>/', views.UserDetailView.as_view()),
11     path('verifyToken/', views.VerifyTokenView.as_view()),
12
13 ]
```

Container e Imagen

Se creó el microservicio en Dockerfile, por lo tanto, teniendo Dockerfile abierto, desplegó el servicio en heroku, y se entró en el contenedor de heroku y se hizo un push en este contenedor externo, este proceso se refleja a continuación:

Se crear la imagen:



The screenshot shows the Visual Studio Code interface with the Dockerfile editor open. The Dockerfile content is as follows:

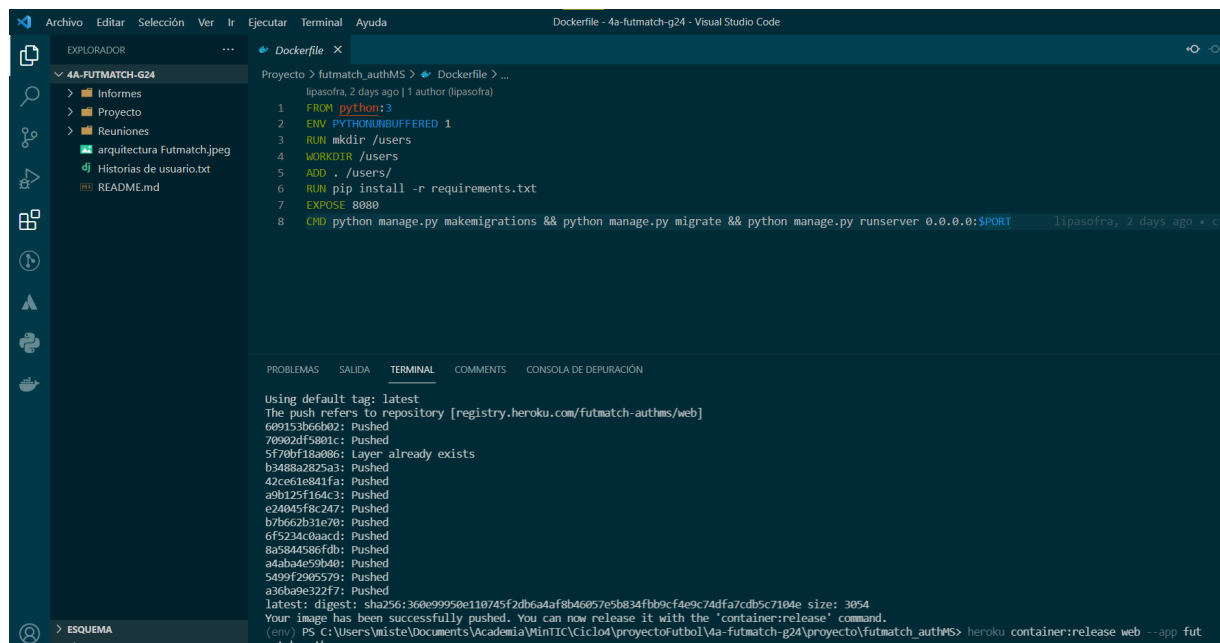
```
1 FROM python:3
2 ENV PYTHONUNBUFFERED 1
3 RUN mkdir /users
4 WORKDIR /users
5 ADD . /users/
6 RUN pip install -r requirements.txt
7 EXPOSE 8080
8 CMD python manage.py makemigrations && python manage.py migrate && python manage.py runserver 0.0.0.0:$PORT
```

The terminal output shows the following steps:

```
=> => exporting layers
=> => writing image sha256:4c6de8fdb7ac606492236cb256f22969bd792f1ee0e361680056bee4a1d80126
=> => naming to registry.heroku.com/futmatch-authms/web

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
=== Pushing web (C:\Users\miste\Documents\Academia\MinTIC\Ciclo4\proyectoFutbol\4a-futmatch-g24\proyecto\futmatch_authMS\Dockerfile)
Using default tag: latest
The push refers to repository [registry.heroku.com/futmatch-authms/web]
609153b66b02: Pushed
70902df5801c: Pushed
5f70bf18a086: Layer already exists
b3488a2825a3: Pushed
42ce61e841fa: Pushed
a9b125f164c3: Pushed
e24045f8c247: Pushed
b7b662b31e70: Pushed
6f5234c0aacd: Pushing [=====>] 126.6MB/528.4MB
8a5844586fdb: Pushing [=====>] 113.8MB/151.9MB
a4aba4e59b40: Pushed
5499f2905579: Pushed
a36ba9e322f7: Pushing [=====>] 70.62MB/123.9MB
```

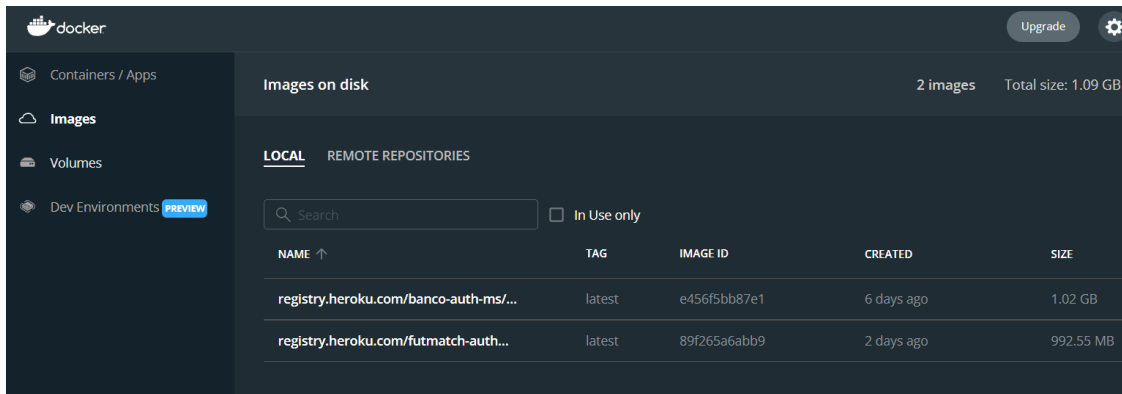
Se realiza el despliegue:



The screenshot shows the Visual Studio Code interface with the Dockerfile editor open. The Dockerfile content is the same as in the previous screenshot. The terminal output shows the following steps:

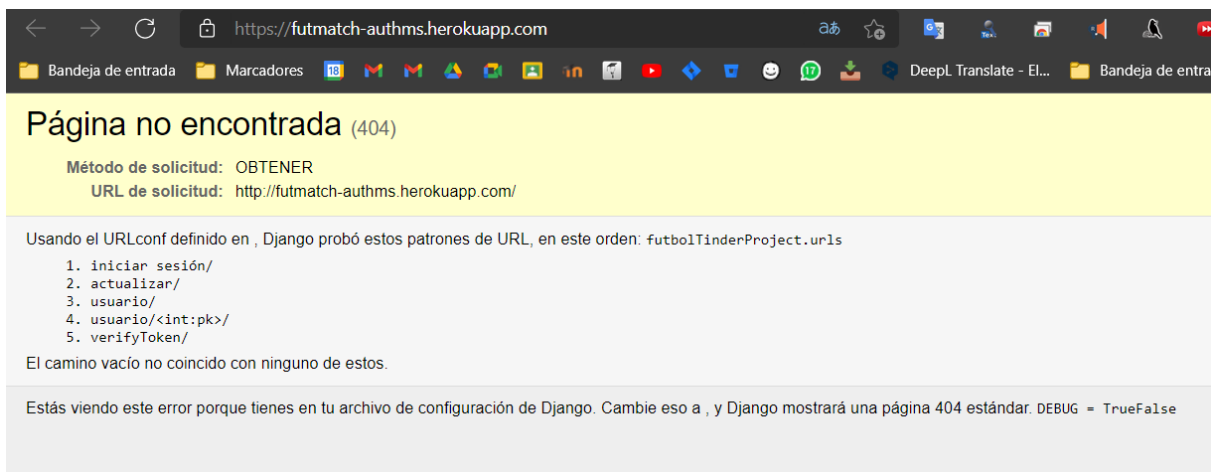
```
Using default tag: latest
The push refers to repository [registry.heroku.com/futmatch-authms/web]
609153b66b02: Pushed
70902df5801c: Pushed
5f70bf18a086: Layer already exists
b3488a2825a3: Pushed
42ce61e841fa: Pushed
a9b125f164c3: Pushed
e24045f8c247: Pushed
b7b662b31e70: Pushed
6f5234c0aacd: Pushed
8a5844586fdb: Pushed
a4aba4e59b40: Pushed
5499f2905579: Pushed
a36ba9e322f7: Pushed
latest: digest: sha256:36e99950e118745f2d6c4a4f8b46957c6b34fbbcf469c74df37cd05c7184e size: 3054
Your image has been successfully pushed. You can now release it with the 'container:release' command.
(ew) PS C:\Users\miste\Documents\Academia\MinTIC\Ciclo4\proyectoFutbol\4a-futmatch-g24\proyecto\futmatch_authMS> heroku container:release web --app futmatch-authms
```

Se genera la imagen en Dockerfile:



App en Heroku

Se desplegó la aplicación en Heroku, generando una URL para realizar las respectivas pruebas con Postman



Pruebas (postman)

Después del despliegue de la aplicación en Heroku realizamos las respectivas pruebas en el aplicativo web de Postman.

Primero registramos un usuario nuevo, con su respectiva información:





RuedaGJUAN / 4a-futmatch-g24

Public

Watch

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

main

Commits on Nov 18, 2021

main Merge branch 'develop'

lipasofra committed 9 minutes ago

cf73f1f

Commits on Nov 16, 2021

creacion dockerfile

lipasofra committed 2 days ago

dd39748

nueva carpeta authMS y migraciones

lipasofra committed 2 days ago

c9c4882

Commits on Nov 12, 2021

organizo en carpetas

lipasofra committed 6 days ago

eeffb66

se agrega inform sprint 1 y arquitectura

lipasofra committed 6 days ago

3adeb47

Historias de usuario

lipasofra committed 6 days ago

93c7a2c

se agregaron los integrantes

lipasofra committed 6 days ago

16853e9

Commits on Nov 8, 2021

doc de reunion 2

lipasofra committed 10 days ago

767a533