Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ciencias de la Computación e Informática

CI0128 – Proyecto Integrador: Ingeniería de Software y Bases de Datos
Profesoras: Rebeca Obando y Alexandra Martínez

Reporte: Sprint 3

Nombre del Equipo: Ta' Bueno

Miembros:
Emmanuel D. Solís - B97670 (Scrum Master)
emmanuel.solispomares@ucr.ac.cr
Gabriel Zúñiga - B98755
gabriel.zunigaorozco@ucr.ac.cr
Jan Murillo - B95447
jan.murillo@ucr.ac.cr
Kevin Arguedas - B80626
kevin.arguedasmuriel@ucr.ac.cr
Luis D. Chinchilla - B82227
luis.chinchillaotarola@ucr.ac.cr

Índice

1. Código Fuente del Sistema

Se encuentra tanto adjunto en esta carpeta compromida como en el enlace del GitHub previamente compartido.

2. Code Review

A realizarse según nuestra cita para el día viernes 22 de Julio a la 1:30pm.

3. Unit Testing

Proyecto de pruebas se encuentra en la carpeta */Planilla/planilla_backend_testing*, se adjuntan pruebas para todos los siguientes métodos que fueron todos los métodos públicos que según el enunciado representaban funcionalidad que había que implementar:

3.1. Dashboard Controller

- `GetDashboard`
- `GetDashboardEmployee`

3.2. Deductions Controller

- `VoluntaryDeductionsBeingUsedByEmployee`
- `VoluntaryDeductionsNotBeingUsedByEmployee`
- `EstablishVoluntaryDeductionStatus`
- `UpdateVoluntaryDeductionEmployee`

3.3. Payment Controller

- `GetEmployeePaymentHistory`

3.4. Projects Controller

- `GetLastPayment`

3.5. Report Controller

- `GetProjects`
- `GetReport`

3.6. User Controller

- `GetHours`
- `ManageHours`

4. UI Testing

Videos de prueba de ejecución de las pruebas, junto con el respectivo reporte de los **casos de prueba**, ademas del proyecto que crea *Selenium* se encuentran en la carpeta de este directorio: ***ui_testing***.

Cabe mencionar de forma relevante que se hicieron las pruebas **grabándolas en el navegador**, no por código, según el enunciado permitía la libertad de ejecutar cualquiera de las dos formas.

5. Deployment de la aplicación en Azure

Escogimos la plataforma **Azure** para el despliegue de la aplicación dado que es la que mayormente conocemos y donde también teniamos el uso de créditos gratis para poder hacerlo.

Algo importante de estipular es que nuestro repositorio de GitHub originalmente tiene en el los tres proyectos: archivos de la base de datos, backend y frontend. Por lo que fue necesario crear dos repositorios copias de estos por aparte, privados, para tener en cada uno de forma separada el backend y el frontend, y entonces poder hacer el respectivo *deployment* de cada uno.

Se adjunta en cada seccción los enlaces respectivos para que se pueda ver que el sistema sí esta accesible.

5.1. Base de Datos

Para hacer el *deployment* de la base de datos fue crear una **nueva base de datos** en Azure, y usando los scripts que teníamos guardado de la creación de tablas, funciones y procedimientos, además de los índices y transacciones; pudimos crear la nueva base de datos sin problema alguno.

Enlace del servidor de base de datos: payroll-database-emmasolis.database.windows.net. Por favor tengase en cuenta que el servidor de base de datos está en una zona de Azure que por motivos de seguridad exije que a los usuarios les esté registrada la IP desde la que va a acceder por lo que probablemente no le permita conectarse pero con gusto puedo reunirme con usted y mostrarle que si está activa.

Tambien se adjunta el **Connection String**: *Server=tcp:payrolldatabaseemmasolis.database.windows.net,1433;Initial Catalog=payroll_database;Persist Security Info=False;User ID=TaBuenoAdmin;Password=UCRecci2022$;*.

Primero tuvimos que crear un servidor de base de datos, y luego entonces crear la base de datos para el proyecto. En la Figura 1 podemos ver el estado actual de dicho servidor.

Una vez que ya teníamos el servidor creamos la base de datos para el proyecto en si, esta la podemos ver en la Figura 2.

Y al tener estas dos cosas ya teníamos la base de datos en Azure para luego poder ser usada por las otras dos fases de la aplicación: *Backend* y *Frontend*.

5.2. Backend

Para hacer el *deployment* del backend fue más sencillo pues solo fue necesario tomar el repositorio separado del backend y en Azure crear un nuevo *App Service* el cuál se le configura que es un .NET API y como esta conectado por medio de GitHub, Azure la conexión y *deployment* de forma automática por debajo. Con eso ya tenemos tanto la base de datos como el backend (API) listos y en linea, faltando solo el frontend.

Enlace del servidor del Backend: https://payroll-api-tabueno.azurewebsites.net.

Podemos ver la vista general del backend en la Figura 3.

5.3. Frontend

En lo que respecta al frontend fue donde tuvimos multiples problemas y fue la unica parte que no logramos poner en linea del todo, despues de toda una serie de errores llegamos a un punto donde la aplicaciónn sí se subia correctamente sin embargo el servidor no lograba identificar cuaál era el puerto que Azure le estaba asiganando a la aplicación, y por lo tanto tomaba el puerto por defecto que era el 3000 pero esto hacia que no puediera accederse.

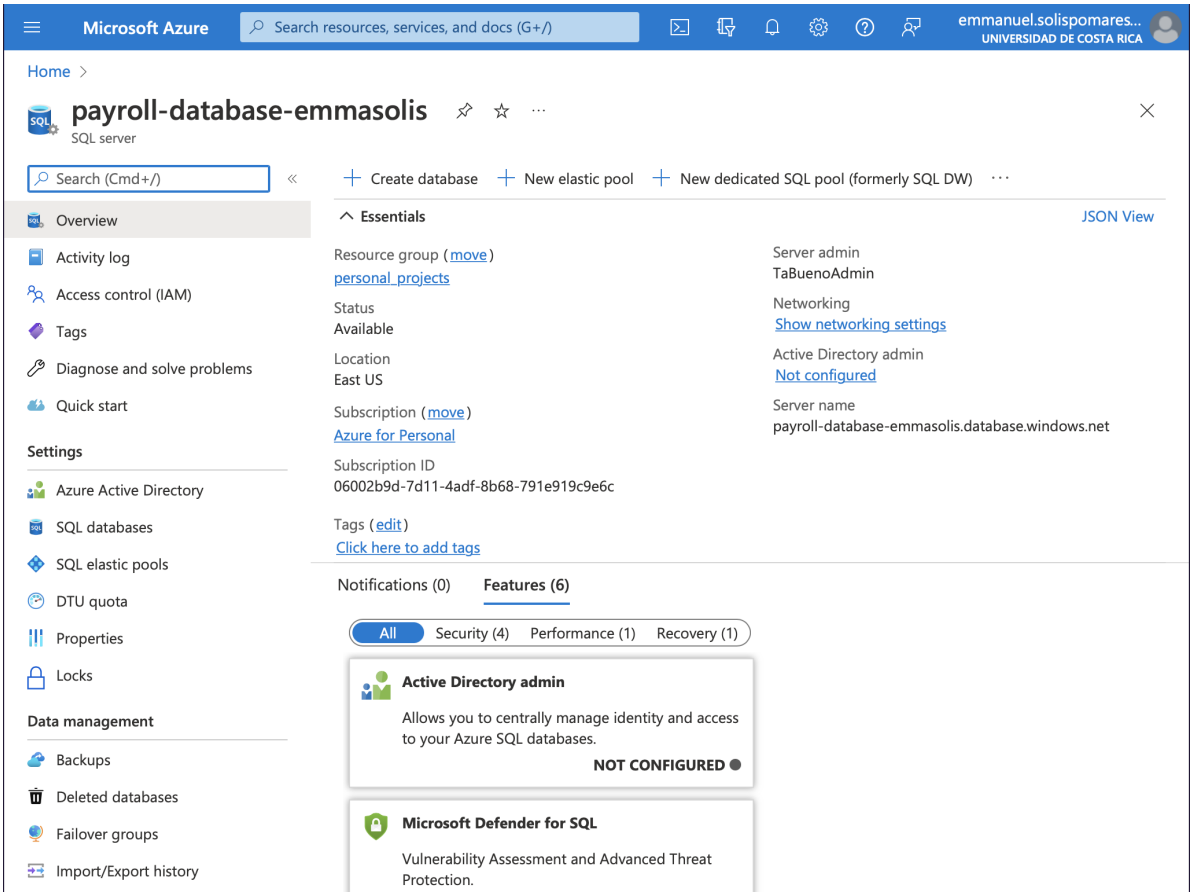

Figura 1: Vista general del servidor de base de datos en Azure.

Enlace del servidor del Frontend: https://payroll-tabueno.azurewebsites.net.

Podemos ver en la Figura 4 que efectivamente el enlace hacia conexión con el App Service pero al no encontrar el servidor corriendo en el puerto que debía entonces no devolvía nada.

De forma similar podemos ver en la Figura 5 que los deployments de la aplicación en Azure se estaba haciendo de forma correcta y que incluso pasaba sin errores, comprobando así que efectivamente el único problema era que el sistema no estaba escuchando en el puerto que debía.

5.3.1. Problemas encontrados al subir el Frontend

- Habían **archivos que evitaban que el servidor compilaba** porque el nombre de una carpeta de donde se traían componentes estaba con el **mismo nombre pero una letra en mayúscula en vez de minúscula**, este error no había sido detectado antes porque se corría en Windows y el Windows como no distingue entre mayúsculas y minúsculas, entonces no se detectaba el error. Este errror nos consumió bastante tiempo para poder ser encontrado.
- El **script del deployment no funcionaba** correctamente, tenía ciertas operaciones que hacía cada deployment increiblemente largo, aproximadamente 2 horas, por lo que hasta el más pequeño cambio en el frontend hacia que tuvieramos que esperar 2 horas extras.
- Se intentó hacer la **IP de la casa de Emmanuel como publica** como un método alternativo para hacer el sistema accesable a todo el internet, sin embargo este redireccionamiento no dió resultado,

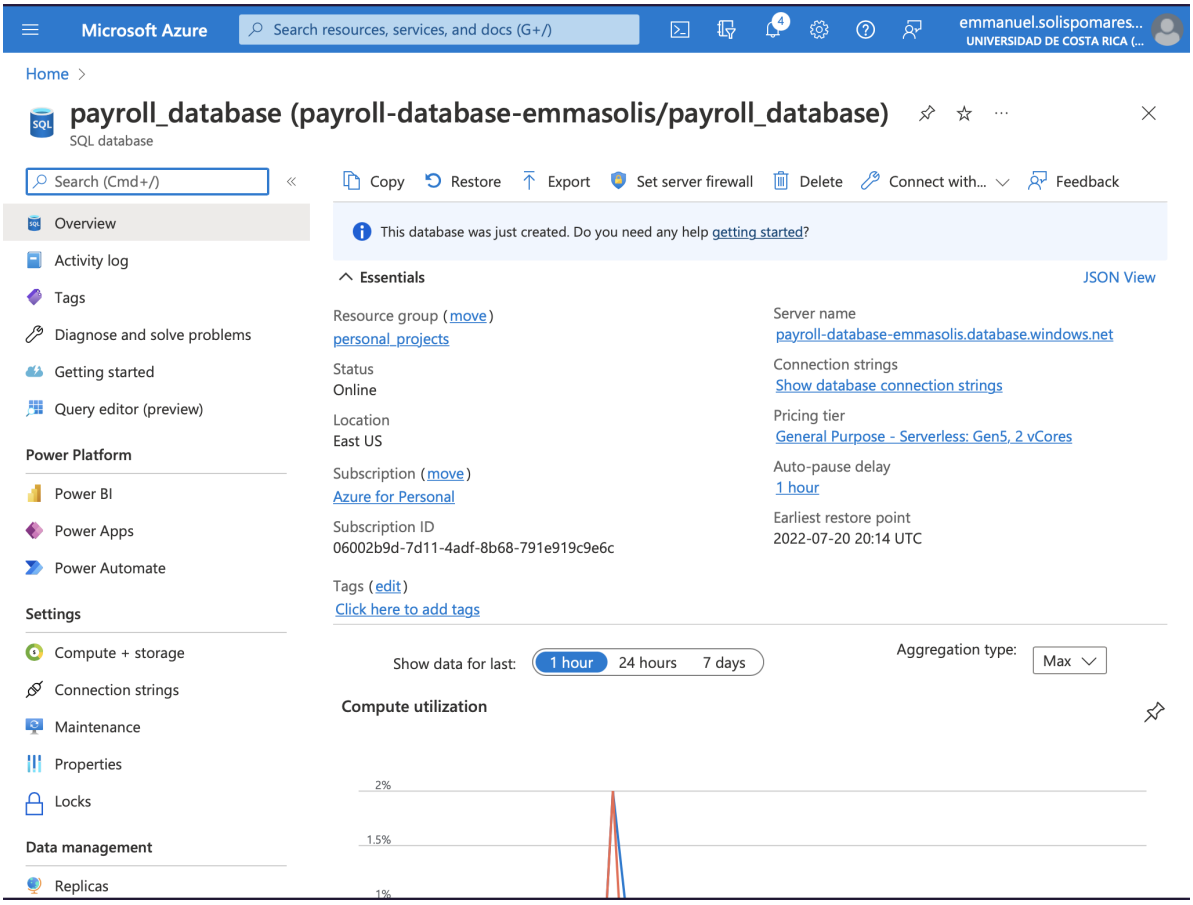

Figura 2: Vista general del de la base de datos *payroll* en Azure.

principalmente porque las IP públicas tienden a cambiar mucho y muy seguido, y porque los certificados SSL para conexiones HTTPS se mostraban como inseguros.

- El compañero **gabriel también intentó hacer una IP pública** para el frontend, pero no se pudo dado que salía el certificado SSL como invalido.
- Al final se logró subir la aplicación a Azure pero el **servidor no detectaba cuál era el puerto que debía tomar** por lo que por defecto tomaba el 3000, pero ese no era el que estaba escuchando el enlace de Azure por lo que aunque el servicio estaba en linea y sin errores no se lograba acceder a esta misma.

6. Sprint Backlog

Se encuentra en el siguiente enlace de la profesora Rebeca Obando: `https://ingesoftg001.atlassian.net/jira/software/projects/TB/boards/5/backlog`. Con todos los requerimientos solicitados según el enunciado.

7. Entregable de las ceremonias

Se encuentra adjunto en el archivo de esta carpeta llamado **ceremonias_scrum.pdf**. Ademas se adjunta el video respectivo de la reunión *Sprint Retrospective* con el nombre **retrospective_meeting.mp4**. También

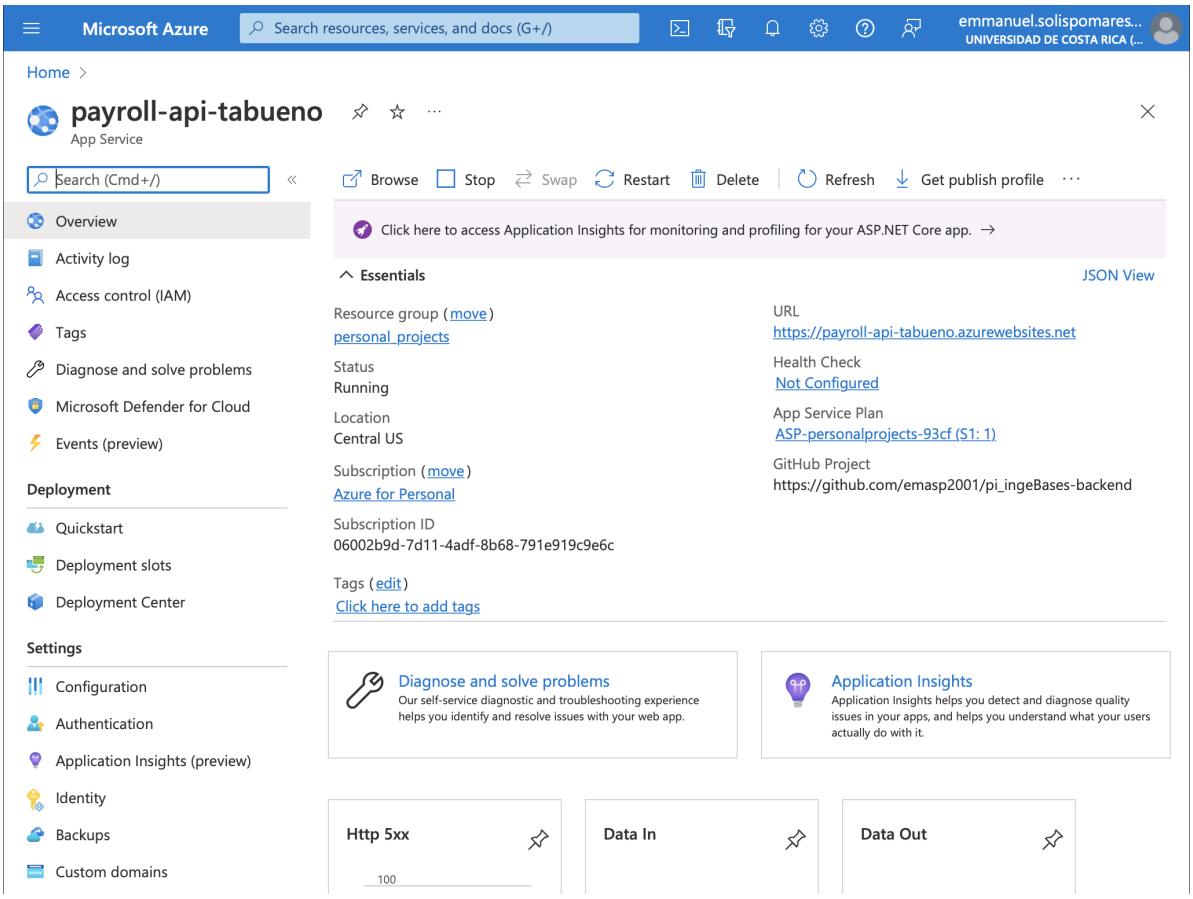

Figura 3: Vista general del Backend API en Azure.

tenemos el gráfico **Sprint burndown chart** en la Figura 6.

8. Mock Ups

Se encuentran en el archivo *mock_ups.pdf*. Para este sprint 3 correspondían los *Mock Ups* de los reportes y los dashboard tanto del empleado como del empleador.

9. Implementación de la Base de Datos

Nuestra base de datos *Ta Bueno* se encuentra implementada con las tablas que teníamos en los diagramas de Modelo EntidadRelación y Lógico que fueron previamente presentados y aprobados por la profesora Alexandra.

Se agregaron las respectivas funcionalidades de los índices y transacciones, y todo lo necesario según esta entrega; cabe mencionar que la base de datos ha sido ampliamente presentada para revisión a la profesora, y la profesora ha aprobado el diseño e implementación que hemos tenido.

Se adjuntan a demás todos los *scripts* de todo los creado en la base de datos, este se encuentra en la carpeta **database_scripts**. Cabe mencionar que la base de datos que se enceuntra en Azure es una replica exacta de esta base de datos por lo que con revisar esta que se encuentra en los servidores de la *Escuela de Ciencias de la Computación e Informática* es suficiente.

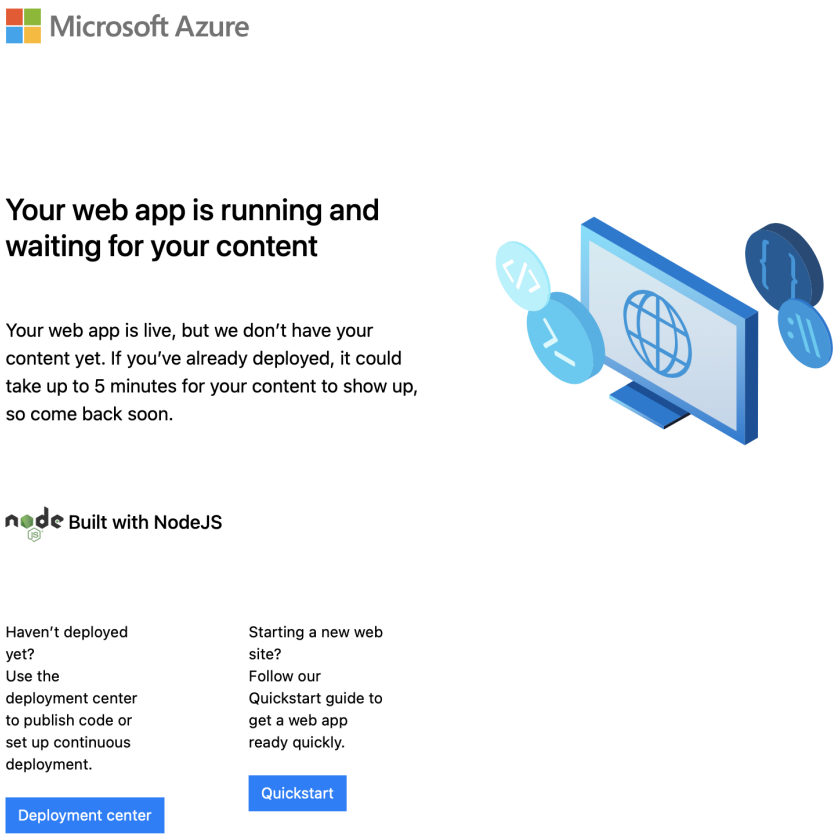

Figura 4: Vista general del Frontend en Azure.

10. Presentación durante Sprint Review

Se realizará la exposición el día 27 de Julio del 2022 debido a problemas técnicos con la base de datos fuera de nuestro alcance.

11. Autoevaluación y Coevaluación

Cada miembro del equipo se encarga individualmente de enviar su documento de auto y coevaluación.

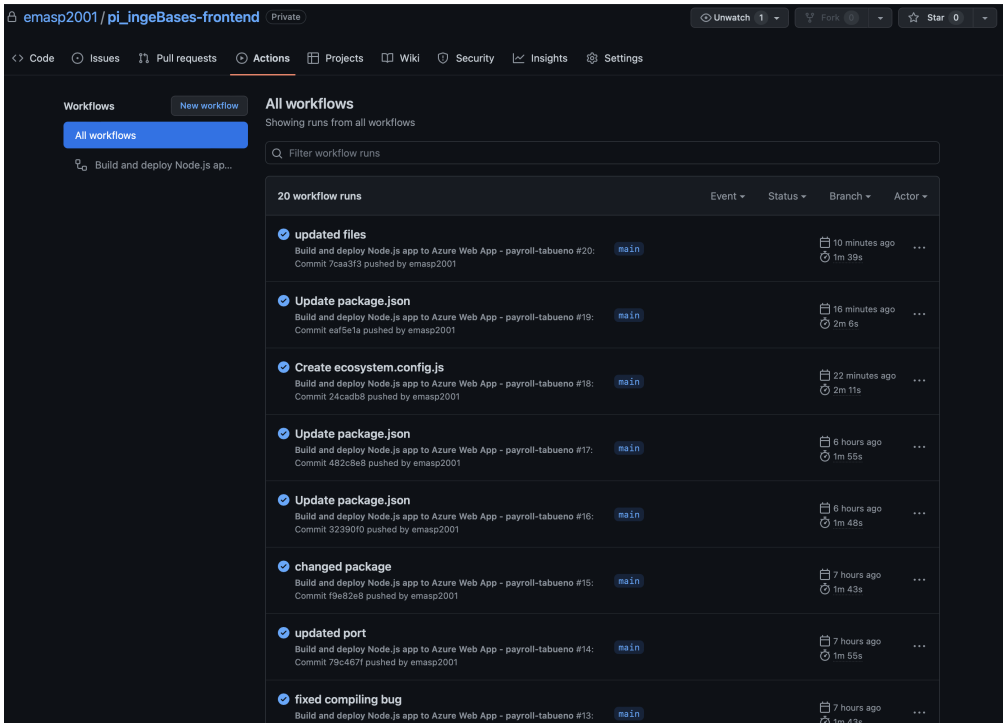

Figura 5: Git Deploys del Frontend.

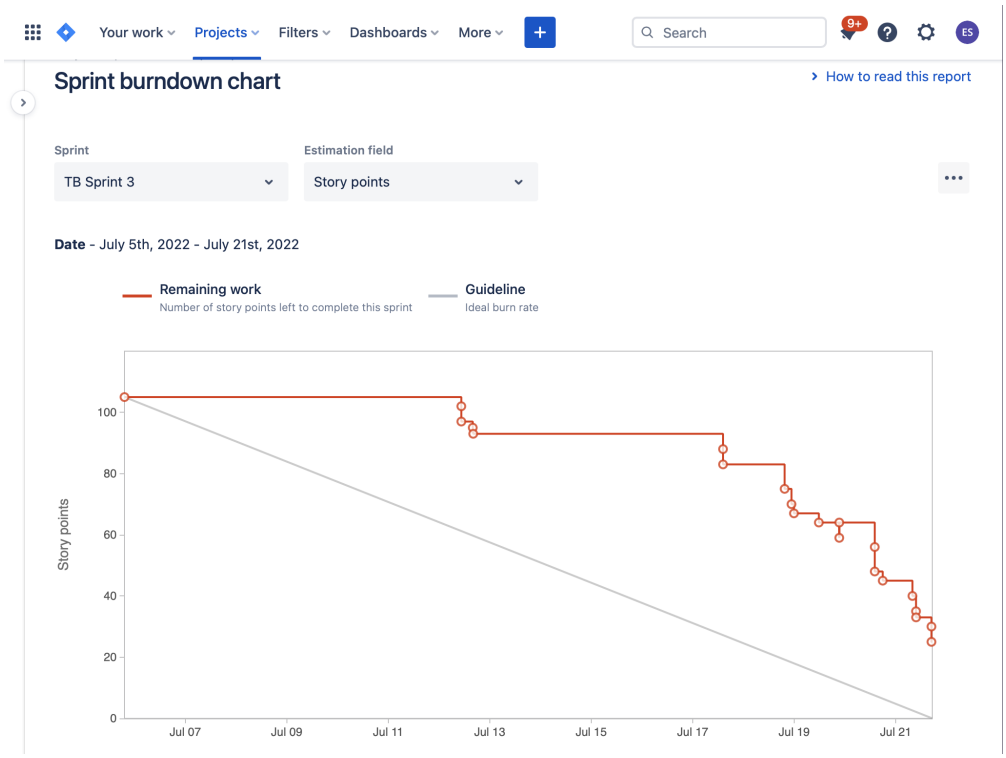

Figura 6: Gráfico de Burndown del Sprint 3.