

Universidad ORT Uruguay

Facultad de Ingeniería

Arquitectura de Software en la Práctica

Obligatorio 1

Matías Hernández 169236
Diego Klappenbach 178226
Esteban Muzio 92391

Docente: Pablo Vilas

2019

Índice

Índice	2
1. Introducción	4
2. Antecedentes	4
2.1 Propósito del sistema	4
3. Descripción de la arquitectura	5
3.2 Vista de componentes y conectores	5
3.2.1. Representación primaria	5
3.2.2. Catálogo de elementos	6
3.2.3. Interfaces	6
3.2.5. Decisiones de diseño	7
3.3 Vista de despliegue	8
3.3.1. Representación primaria	8
3.3.2. Catálogo de elementos	9
3.3.3. Relación con componentes	10
3.3.4. Decisiones de diseño	10
4. Justificaciones de diseño	11
4.1 Versiones de Ruby y Ruby on Rails	11
4.2 Entorno de desarrollo	11
4.3 Librerías de terceros (Gemas)	12
4.4. Justificación de los Requerimientos No Funcionales	13
4.4.1. RNF1: Performance	14
4.4.2. RNF2: Confiabilidad y disponibilidad	14
4.4.3. RNF3: Configuración y manejo de secretos	14
4.4.4. RNF4: Autenticación y autorización	15
4.4.5. RNF5: Seguridad	15
4.4.6. RNF6: Código fuente	16
4.4.7. RNF7: Pruebas	18
4.4.8. RNF8: Identificación de fallas	18
4.5. Justificación de los Requerimientos Funcionales	19
4.5.1. RF1: Registro de usuario	19
4.5.2. RF2: Autenticación de usuario	21
4.5.3. RF3: Gestión de clave de aplicación	21
4.5.4. RF4: Gestión de promociones	21
4.5.5. RF5: Listado de promociones	24
4.5.6. RF6: Baja de promoción	24
4.5.7. RF7: Reporte de uso	25

4.5.8. RF8: Evaluación de promoción	25
5. Proceso de desarrollo, testing y deployment	26
5.1. Instrucciones para el desarrollo:	26
5.2. Instrucciones para las pruebas:	27
5.3. Instrucciones para el deploy:	27
5.4. Gestionar la app en producción:	29
6. Cumplimiento de The Twelve-Factor App	29
7. Pruebas	32
7.1. Pruebas automáticas	32
7.2. Pruebas de carga	35
8. Gestión de proyecto	39
8.1. Metodología	39
8.2. Herramientas	40
8.3. Gestión del Alcance	42
9. Detalle de Endpoints	43
10. Descripción de la Interfaz de Usuario	46

1. Introducción

El siguiente documento detalla la arquitectura diseñada para el nuevo servicio Coupons, el cual pretende ser el servicio de cupones y descuentos referente en la región.

Se describen en detalle las decisiones arquitectónicas y de diseño tomadas, analizando la solución de software desde diversos puntos de vista, y resaltando sus características más destacables.

Todas las justificaciones de diseño van acompañadas de argumentos sólidos, los cuales se vinculan (cuando ello es posible), con las tácticas o patrones de arquitectura de software aplicados en cada caso.

Por último, se describen las etapas necesarias para el despliegue de toda la solución, contemplando todos estos pasos a suficiente nivel de detalle como para que el producto tenga alta probabilidad de quedar operativo sin mayores contratiempos y sin afectar la disponibilidad del servicio.

2. Antecedentes

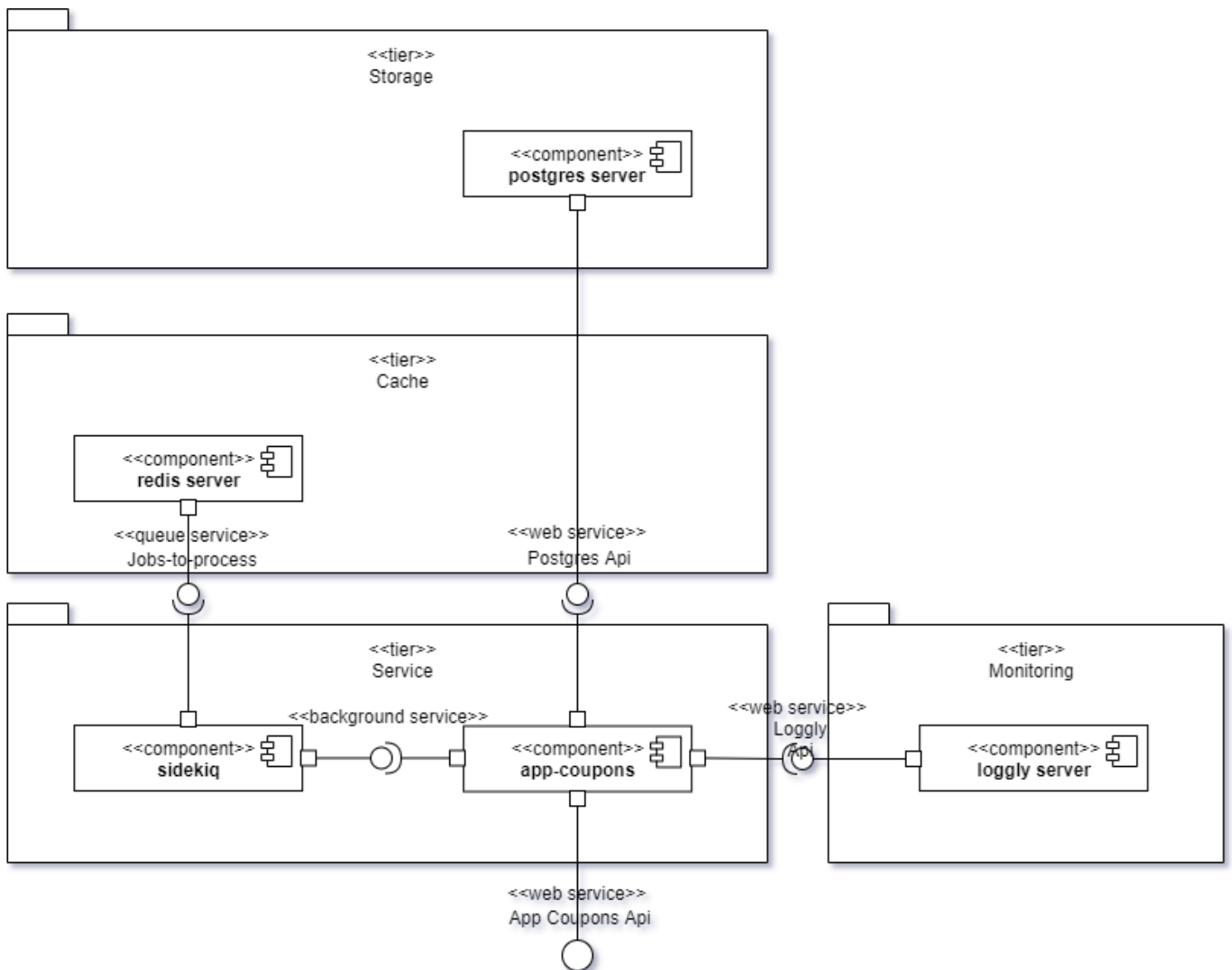
2.1 Propósito del sistema

Coupons, surge a partir de una necesidad común en el mundo del e-commerce, el poder crear, gestionar y realizar la trazabilidad de cupones promocionales y descuentos, todo ello desde un mismo lugar. Esta solución intentará revolucionar el mercado, ofreciendo un producto moderno y fácil de utilizar, el cual será ofrecido a sus clientes como servicio (SaaS).

3. Descripción de la arquitectura

3.2 Vista de componentes y conectores

3.2.1. Representación primaria



3.2.2. Catálogo de elementos

Componente/ Conector	Tipo	Descripción
<i>postgres server</i>	<i>Postgres Server Api</i>	<i>Servicio web que actúa como interfaz para poder acceder a la base de datos Postgres.</i>
<i>redis server</i>	<i>Redis Server Api</i>	<i>Servicio de base de datos en memoria ram, clave valor, utilizada para encolar jobs que serán procesados en background.</i>
<i>sidekiq</i>	<i>Background job application</i>	<i>Servicio procesador de background jobs en diferentes threads al principal.</i>
<i>app-coupons</i>	<i>Web Server Api</i>	<i>Servicio proveedor de servicios principales de la aplicación.</i>
<i>loggly server</i>	<i>Logging Server Api</i>	<i>Servicio de logs, que gestiona y centraliza los logs provenientes del stdout de app-coupons.</i>

3.2.3. Interfaces

Interfaz	<i>Postgres Api</i>
La implementa	<i>postgres server</i>
Servicio	Descripción
<i>Api para manejar PostgreSQL.</i>	<i>Es utilizado por app-coupons para guardar y recolectar los datos que necesitan ser persistidos de la app-coupons.</i>

Interfaz	<i>jobs-to-process</i>
La implementa	<i>redis server</i>
Servicio	Descripción
<i>Api para manejar Redis.</i>	<i>Es utilizado por sidekiq para encolar jobs que realiza en background.</i>

Interfaz	<i>Sidekiq Service</i>
----------	------------------------

La implementa	<i>sidekiq</i>
Servicio	Descripción
<i>Api para manejar background jobs.</i>	<i>Es utilizado por app-coupons para gestionar los background jobs de la aplicación.</i>

Interfaz	<i>Loggly Api</i>
La implementa	<i>loggly server</i>
Servicio	Descripción
<i>Api para guardar logs.</i>	<i>Es utilizado por app-coupons para guardar los logs del stdout de la aplicación.</i>

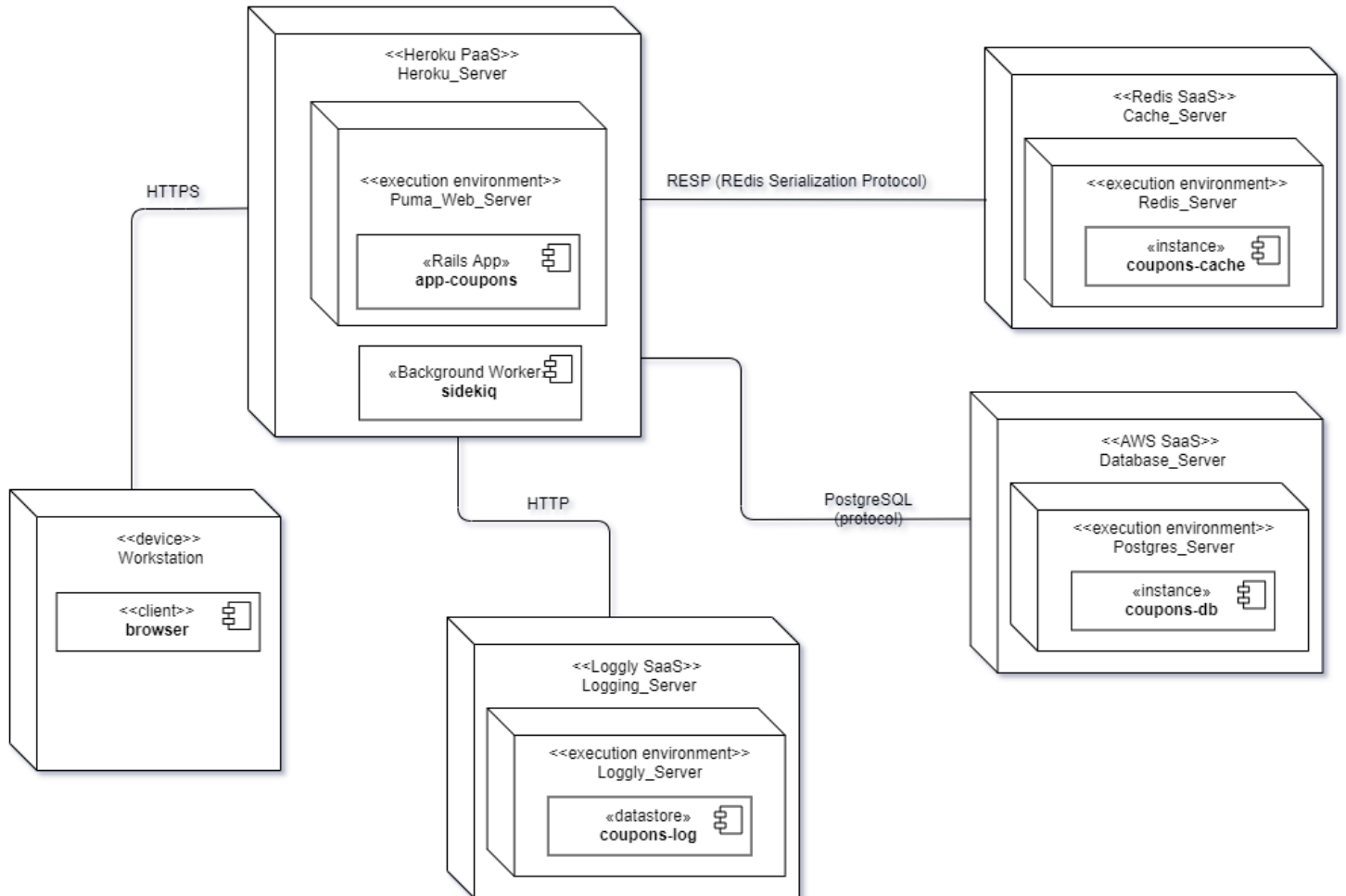
Interfaz	<i>App Coupons Api</i>
La implementa	<i>app-coupons</i>
Servicio	Descripción
<i>Evaluación de promoción.</i>	<i>Pre-requisito: Se debe contar con una clave de aplicación válida.</i> <i>Es utilizado para evaluar si corresponde aplicar una promoción o no.</i>
<i>Reporte de promoción.</i>	<i>Pre-requisito: Se debe contar con una clave de aplicación válida.</i> <i>Es utilizado para ver el reporte de una promoción.</i>

3.2.5. Decisiones de diseño

Las justificaciones de diseño se encuentran en [Justificaciones de diseño](#).

3.3 Vista de despliegue

3.3.1. Representación primaria



3.3.2. Catálogo de elementos

Nodo	Características	Descripción
<i>Heroku_Server</i>	<i>512 MB RAM, 1 Worker</i>	<i>Servidor Heroku que aloja a app-coupons y se conecta con diferentes servicios.</i>
<i>Puma_Web_Server</i>	<i>N/A</i>	<i>Servidor Web que recibe las peticiones en los Endpoints de la aplicación app-coupons.</i>
<i>Cache_Server</i>	<i>N/A</i>	<i>Servidor de caché, perteneciente a Redis-To-Go, que contiene el servidor de Redis utilizado por nuestro sistema.</i>
<i>Redis_Server</i>	<i>5,2 MB, 1 database</i>	<i>Servidor de Redis utilizado por sidekiq para persistir temporalmente los background jobs.</i>
<i>Database_Server</i>	<i>N/A</i>	<i>Servidor de base de datos de AWS que contiene el servidor de PostgreSQL utilizado por nuestro sistema.</i>
<i>Postgres_Server</i>	<i>10.000 filas, 20 conecciones</i>	<i>Servidor PostgreSQL usado para persistir los datos de la aplicación app-coupons.</i>
<i>Logging_Server</i>	<i>N/A</i>	<i>Servidor de Logging, perteneciente a Loggly, que contiene el servidor de Loggly utilizado por nuestro sistema.</i>
<i>Loggly_Server</i>	<i>Volumen diario de 200MB, Retención por 7 días.</i>	<i>Servidor de Loggly usado por app-coupons para registrar los logs del stdout.</i>

Conector	Características	Descripción
<i>HTTP</i>	<i>80/TCP</i>	<i>Se utiliza el conector HTTP para enviar los logs de app-coupons a Loggly.</i>
<i>HTTPS</i>	<i>443/TCP</i>	<i>Se utiliza el conector HTTPS para que los clientes puedan acceder a través de un protocolo seguro a los endpoints de app-coupons.</i>
<i>RESP</i>	<i>9530/TCP</i>	<i>Cientes Redis se comunican con Redis Server mediante este protocolo.</i>
<i>PostgreSQL</i>	<i>5432/TCP</i>	<i>Cientes PostgreSQL se comunican con el servidor PostgreSQL mediante este protocolo.</i>

3.3.3. Relación con componentes

Nodo	Componente
<i>Heroku_Server</i>	<i>app-coupons</i> <i>sidekiq</i>
<i>Puma_Web_Server</i>	<i>app-coupons</i>
<i>Cache_Server</i>	<i>redis-server</i>
<i>Redis_Server</i>	<i>redis-server</i>
<i>Database_Server</i>	<i>postgres-server</i>
<i>Postgres_Server</i>	<i>postgres-server</i>
<i>Logging_Server</i>	<i>loggly-server</i>
<i>Loggly_Server</i>	<i>loggly-server</i>

3.3.4. Decisiones de diseño

Las justificaciones de diseño se encuentran en [Justificaciones de diseño](#).

4. Justificaciones de diseño

4.1 Versiones de Ruby y Ruby on Rails



Dentro de los requerimientos no funcionales, encontramos uno (*RNF6*) que refiere al lenguaje de programación y al framework que debe utilizarse para la implementación de la solución. Concretamente el lenguaje solicitado es Ruby, y el framework para todo el desarrollo web de esta solución es Ruby on Rails. En ambos casos no se hace referencia alguna a las versiones que deben utilizarse, por lo cual ello queda a criterio exclusivo por parte del equipo de desarrollo.

Dadas las características del sistema, la falta de experiencia dentro del equipo de desarrollo en la utilización tanto de Ruby como de Ruby on Rails, y sumado al hecho que se espera tener rápidamente desplegada una versión simplificada en producción, ya que se han cerrado acuerdos con cinco clientes, fue que se optó por escoger las últimas versiones estables tanto de Ruby como de Rails, intentando mitigar problemas por bugs aún no detectados o posibles incompatibilidades con alguna de las gemas más utilizadas. En este sentido fue que se seleccionaron las versiones de Ruby 2.6.4 y la de Rails v6.0.0.

4.2 Entorno de desarrollo



El equipo de desarrollo decidió utilizar Visual Studio Code para toda la implementación del sistema. La elección de este editor, se basó en que se trata de un editor moderno y robusto, el cual cuenta con gran cantidad de plugins para trabajar cómodamente con Ruby y Rails. También otro factor determinante en esta selección, fue que los desarrolladores no trabajan todos sobre las mismas plataformas y este editor soporta muchos sistemas operativos distintos sin requerir

configuraciones y pasos de instalación muy distintos para cada una de ellas. Por otro lado y no menos importante, este es el editor que se utiliza en los ejemplos del curso, por lo cual todo lo que colabore a minimizar las dificultades en la adopción de estas tecnologías nuevas para el equipo, son bienvenidas.

4.3 Librerías de terceros (Gemas)

En la búsqueda por conseguir una implementación de la solución lo más profesional posible e intentar aumentar nuestra productividad durante el proceso de desarrollo, fue que decidimos invertir algo de tiempo previo a la implementación de cualquier nueva característica, intentando descubrir alguna librería (Gema) que nos pudiera facilitar la tarea evitando reinventar la rueda. Todas las librerías que escogimos dentro de la vasta cantidad disponible, son ampliamente utilizadas por la comunidad de desarrolladores Ruby, por lo cual esto nos garantiza que las mismas han sido probadas en una gran cantidad de proyectos, asegurando una buena calidad en su implementación y desempeño.

Para validar la utilización de cada una de estas librerías, nos basamos en los siguientes criterios:

- Investigaciones que realizamos en Internet, leyendo en distintos foros sobre las características y las valoraciones brindadas por otros desarrolladores.
- Análisis de popularidad, cantidad de descargas y nivel de mantenimiento que ropartaba en cada caso la web del servicio de alojamiento de gemas de la comunidad Ruby: <https://rubygems.org>

Gema	Utilidad	Motivo de la elección
<i>rubocop-rails</i>	<i>Formateador y analizador de código estático.</i>	<i>Solicitado como parte del RNF6: Código Fuente.</i>
<i>dotenv-rails</i>	<i>Gestión de variables de ambiente.</i>	<i>Cumplir con el RNF3: Configuración y manejo de secretos. Fácil de utilizar y figuraba como recomendada en un práctico.</i>
<i>annotate</i>	<i>En los Controllers y Models muestra la información del schema.</i>	<i>Facilidad para ver los atributos de cada modelo.</i>
<i>bcrypt</i>	<i>Gestionar password de los usuarios.</i>	<i>Simple y suficiente para el tipo de autenticación requerido en este sistema.</i>
<i>pg</i>	<i>Interfaz con PostgreSQL.</i>	<i>Es la más utilizada para trabajar con PostgreSQL.</i>
<i>acts_as_tenant</i>	<i>Para administrar multi-tenancy.</i>	<i>Vista en clase y muy fácil de utilizar.</i>

<i>bootstrap</i>	<i>Para realizar las vistas con estilos de forma más sencilla.</i>	<i>Cumplir con bases de usabilidad sobre la Interfaz gráfica de usuario del sistema.</i>
<i>jquery-rails</i>	<i>Para poder realizar jquery en la interfaz de usuario.</i>	<i>Cumplir con funcionalidades que aporten al usuario una mejor usabilidad del sistema.</i>
<i>acts_as_paranoid</i>	<i>Para administrar el borrado lógico de las PromotionDefinition.</i>	<i>Cumplir con el RF6: Baja de promoción. Fácil de utilizar.</i>
<i>dentaku</i>	<i>Parseador y evaluador de fórmulas matemáticas y lógicas.</i>	<i>Cumplir con la evaluación de condiciones para el RF4: Gestión de promociones y RF8: Evaluación de promoción.</i>
<i>faker</i>	<i>Generador de fake data.</i>	<i>Genera fácilmente fake data.</i>
<i>factory_bot_rails</i>	<i>Generador de instancias predefinidas</i>	<i>Facilita la creación de las pruebas</i>
<i>has_scope</i>	<i>Permite mapear parámetros desde un controlador a alcances nombrados en los recursos.</i>	<i>Cumplir con los filtros requeridos en el RF5: Listado de Promociones.</i>
<i>image_processing</i>	<i>Permite manejar el procesamiento de imágenes.</i>	<i>Cumplir con la muestra de imagen del RF1: Registro de usuario.</i>
<i>health_check</i>	<i>Verifica que rails está arriba y corriendo, y que se tenga acceso a los recursos correctamente configurados.</i>	<i>Cumplir con el RNF2: Confiabilidad y disponibilidad.</i>
<i>logglier</i>	<i>Envía mensajes de log a Loggly.</i>	<i>Cumplir con el RNF8: Identificación de fallas.</i>
<i>sidekiq</i>	<i>Procesa background jobs.</i>	<i>Ayuda a cumplir con el RNF1: Performance.</i>
<i>jwt</i>	<i>Provee mecanismos para gestionar 'JWT's.</i>	<i>Cumplir con el RNF4: Autenticación y Autorización</i>

4.4. Justificación de los Requerimientos No Funcionales

En esta sección, se especificarán las acciones tomadas y sus justificaciones para cada uno de los requerimientos no funcionales de la especificación de requerimientos.

4.4.1. RNF1: Performance

Ver [7.2. Pruebas de carga](#)

4.4.2. RNF2: Confiabilidad y disponibilidad

Gemas utilizadas:

'Health_check': Brinda un endpoint y las funcionalidades adecuadas para determinar si rails está arriba y corriendo, y si los recursos configurados están respondiendo.

Implementación:

- Devuelve por defecto 200 OK cuando todo está correcto.
- La gema fue configurada en el archivo:
 - './config/initializers/health_check.rb'
- En la ruta estándar prueba que rails está arriba y corriendo, la base de datos postgres y sus migraciones, el gestor de email y cache.

4.4.3. RNF3: Configuración y manejo de secretos

Gemas utilizadas:

'Dotenv-rails': Permite manejar el acceso a variables de entorno.

Implementación:

En producción (Heroku):

- Se ingresaron como variables de entorno dentro del servidor de Heroku las siguientes variables:
 - DATABASE_URL = ...
 - LANG = en_US.UTF-8
 - MAIL_PASSWORD = ...
 - MAIL_ADDRESS = ...
 - RACK_ENV = production
 - RAILS_ENV = production
 - RAILS_LOG_TO_STDOUT = enabled

- `RAILS_SERVE_STATIC_FILES = enabled`
- `SECRET_KEY_BASE = ...`

En desarrollo:

- Se utilizó un archivo que contenía las variables en `'./ .env.development'`.

4.4.4. RNF4: Autenticación y autorización

Gemas utilizadas:

`'Acts_as_tenant'`: Aplica `'multi-tenancy'` sobre los modelos.

Implementación:

Control de acceso basado en roles:

- Para el control de roles se establecieron 2 tipos de usuario:
 - Administrador, que es único para una organización
 - Usuario, que son todos los usuarios de una organización (incluyendo el Administrador).
- Se aplicó `'Acts_as_tenant'` sobre los usuarios, para limitarlos a acceder a datos sobre su organización.
- Cada controller tiene un `'before_action'` que se encarga de autorizar al usuario (dado el permiso que tiene con respecto al endpoint que desea acceder).

`'Application Key's` como `'JWT'`:

- Cuando se genera el `'Application Key'` ([RF:3 Gestión de clave de aplicación](#)) se genera un `'JWT'` con los siguientes atributos:
 - `'Name'`: Nombre del `'Application Key'`
 - `'Codes'`: Array de códigos de las promociones que puede acceder.
- Cuando se recibe un `'Application Key'` tanto para los endpoint de `'Report'` ([RF7: Reporte de uso](#)) o `'Evaluation'` ([RF8: Evaluación de promoción](#)), se desparsea el token que viene en el `'header: Authentication'` correspondiente al `'JWT'` y se puede saber:
 - Su validez, ya que si algo es modificado, no validará con el secreto que fue firmado (esto es inherente a `'JWT'`)
 - Si está autorizado a acceder a la promoción indicada (revisando si la promoción que se desea acceder, está dentro del array de códigos de las promociones).

4.4.5. RNF5: Seguridad

Implementación:

Sistema responde 'HTTP 40X' a cualquier request mal formada:

- Cuando se asigna un 'End-point' que no cumple con las características o está mal formado, se retorna el valor por defecto de error 'HTTP 40X' (dependiendo del error)

Sistema no deja endpoints que no son requeridos:

- Se crearon las rutas y los controladores de manera específica para cada requerimiento del sistema (no se utilizó 'Scaffold')

Comunicación de los componentes de 'Back-end':

- La comunicación con 'Loggly' se hace a través de 'HTTPS' entre el servidor de 'Heroku' y 'Loggly'.
- La comunicación entre el resto de los componentes de 'Back-end' es interna en la red privada de 'Heroku'.

Comunicación entre el 'Front-end' y el 'Back-end':

- La comunicación entre el 'Front-end' y el 'Back-end' está en la red privada de 'Heroku'.

Comunicación entre los clientes y el servidor:

- 'Heroku' provee la comunicación entre el cliente con el servidor a través del protocolo 'HTTPS'.

4.4.6. RNF6: Código fuente

Gemas utilizadas:

'rubocop-rails': Permite verificar el código Ruby para complacer con los estándares.

Implementación:

Lenguaje de programación:

- Se realizó el desarrollo utilizando Ruby como fue solicitado en la especificación.
- Se utilizó el framework Ruby on Rails para las funcionalidades de acceso web.

Código y comentarios en inglés:

- Se respetó este requerimiento, incluyendo el lenguaje inglés en:
 - Código fuente
 - Comentarios del código fuente
 - 'Commit's de Github.
 - Nombre de los 'branch' de Github.
 - 'README.md' de Github.
- Se utilizó la gema rubocop para validar que se esté respetando la guía de estilos de Ruby.

Control de versiones:

- Se utiliza el repositorio GIT, en GitHub de la organización de Arquitectura de Software en la Práctica para el desarrollo del proyecto.
- Se creó 'README.md' que contiene:
 - Propósito
 - Alcance
 - Dependencias externas
 - Instrucciones de desarrollo
 - Instrucciones de testing
 - Instrucciones de deployment
 - Instrucciones para gestionar la aplicación en producción

Manejo de branches:

- Se utilizó Git-Flow.
 - Una rama principal 'master', la cual contiene los release.
 - Una rama 'develop', la cual contiene los desarrollos que fueron previamente aprobados por un 'pull-request'.
 - Se crea una nueva rama por cada feature, fix o enhancement, que una vez terminado el desarrollo de la misma, se hace un 'pull-request' para mergearla con 'develop'.

4.4.7. RNF7: Pruebas

Implementación:

Pruebas automatizadas para requerimientos funcionales:

Ver [7.1. Pruebas automáticas](#)

Planes de prueba de carga:

Ver [7.2. Pruebas de carga](#)

4.4.8. RNF8: Identificación de fallas

Para la identificación de fallas, se utilizó como 'Software as a Service (SaaS)' [Loggly](#), ya que provee tanto una poderosa interfaz como una captación de syslogs en tiempo real de la aplicación. Los logs son mantenidos por más de 24 hrs.

Gemas utilizadas:

'Logglier': Envía mensajes a Loggly usando Syslog/UDP.

Implementación:

En desarrollo:

- Se decidió utilizar un conector a Loggly desde un contenedor Docker para facilitar el uso de la misma desde plataformas diferentes de desarrollo (Mac, Ubuntu y Mint).
- Comando para inicializar el contenedor de docker:
 - `sudo docker run -d -p 514/udp --name loggly-docker -e TOKEN=TOKEN -e TAG=Docker sendgridlabs/loggly-docker`
- Luego se agregó, la configuración necesaria para que funcionara en `config/environments/.rb`:
 - `require 'logglier'`
 - `config.logger = Logglier.new("https://logs-01.loggly.com/inputs/TOKEN/tag/ruby/", :threaded => true)`

En producción:

- Heroku provee una forma simplificada de exportar los logs del sistema a Loggly, por lo tanto, se realizó de la siguiente forma:

- `heroku drains:add http://logs-01.loggly.com/bulk/TOKEN/tag/heroku --app HEROKU_APP_NAME`

4.5. Justificación de los Requerimientos Funcionales

4.5.1. RF1: Registro de usuario

Gemas utilizadas:

'image_processing': Provee procesamiento de imagen a alto nivel.

'acts_as_tenant': Provee capacidad de 'Multi-Tenancy' a los modelos indicados.

'bcrypt': Provee capacidad encriptar contraseñas.

Organization:

Organización que actuará como tenant para todos los usuarios que pertenecen a la misma.

- 'id': Identificador único que internamente es usado para identificar una organización.
- 'name': Nombre de la organización, debe ser único.
- 'admin_id': Id del administrador de la organización.

User:

Los usuarios de una organización, son todos los usuarios que pertenecen a la misma, incluyendo a los administradores.

- 'id': Identificador único que internamente es usado para identificar a un usuario.
- 'first_name': Nombre del usuario.
- 'last_name': Apellido del usuario.
- 'email': Email del usuario.
- 'password_digest': Contraseña del usuario encriptada, se utiliza la gema 'bcrypt'.
- 'organization_id': Id de la organización a la que pertenece.

Se le agrega a 'User':

- `acts_as_tenant(:organization)`

Existen 2 modelos que fueron generados automáticamente por ActiveStorage para poder vincular un Avatar a un usuario.

ActiveStorageBlobs

Se encarga de guardar los datos necesarios de la imagen del Avatar en la base de datos.

ActiveStorageAttachments:

Se encarga de vincular un 'ActiveStorageBlob', con un 'User'.

Para generar el link de invitación, se utiliza 'ActionMailer'.

El usuario administrador de la organización, envía por mail a otro integrante no registrado de dicha organización una invitación.

- El sistema crea un usuario temporal, denominado 'temp' con el email del otro integrante.
- Por mail, le llega al integrante invitado, una contraseña con la cual podrá acceder al sistema.
- Una vez dentro, podrá ingresar a editar y podrá ingresar sus datos personales.

4.5.2. RF2: Autenticación de usuario

Pre-requisitos:

- Se requiere que el cliente esté previamente registrado ([RF1: Registro de usuario](#))

Se utiliza un formulario con el email y la contraseña por pantalla.

Una vez el cliente ingresa con los datos, si el usuario está registrado en el sistema, se le dará acceso a la lista de promociones de su organización ([RF5: Listado de promociones](#))

4.5.3. RF3: Gestión de clave de aplicación

Pre-requisitos:

- Previamente a realizar esta acción, el usuario debe haber realizado 'sign in' ([RF2: Autenticación de usuario](#))
- El cliente tiene que ser administrador de la organización.

Gemas utilizadas:

'jwt': Provee mecanismos para gestionar 'JWT's.

Cuando se genera el 'Application Key' se genera un 'JWT' con los siguientes atributos:

- 'name': Nombre del 'Application Key'
- 'codes': Array de códigos de las promociones que puede acceder.

Se utiliza la gema 'jwt' para generar el 'jwt' con los datos provistos.

4.5.4. RF4: Gestión de promociones

Gemas utilizadas:

'dentaku': Parsea y evalúa fórmulas lógicas.

Pre-requisitos:

- Previamente a realizar esta acción, el usuario debe haber realizado 'sign in' ([RF2: Autenticación de usuario](#))
- El cliente tiene que ser administrador de la organización.

Promociones:

Las promociones están divididas en 2 niveles de abstracción:

- Nivel padre, denominado 'PromotionDefinition', que es la definición de la promoción.
- Nivel hijo, denominado 'Promotion', que es cada instancia de un cupón o uso de promoción.

PromotionDefinition:

Cada 'PromotionDefinition' es identificada por un 'id' que es autogenerado.

Contiene los siguientes datos:

- 'id': Autogenerado internamente, es único.
- 'name': Nombre que identifica la promoción en la organización (una misma organización no puede utilizar el mismo nombre para dos promociones diferentes). Por otro lado, dos organizaciones diferentes pueden utilizar el mismo nombre para una promoción, ya que cada organización está aislada del resto.
- 'discount_type': Puede ser 'percentage' or 'value':
 - 'percentage': Representa que la promoción retorna un valor de porcentaje.
 - 'value': Representa que la promoción retorna un valor fijo.
- 'value': Es el valor de retorno de la promoción.
 - Si 'discount_type' es 'percentage', entonces 'value' puede ser un número entero entre 0 y 100.
 - Si 'discount_type' es 'value', entonces 'value' puede ser un número entero mayor a 0.
- 'promotion_state': Representa el estado de la promoción, puede ser 'active' o 'inactive'.
 - 'active': La promoción puede ser evaluada.
 - 'inactive': La promoción no puede ser evaluada.
- 'promotion_type': Representa qué tipo de promoción es, puede ser 'coupon' o 'discount'.
 - 'coupon': La promoción es de tipo coupon, entonces existirán 'n' promociones cupones que referencian esta 'PromotionDefinition'.
 - 'discount': La promoción es de tipo descuento, entonces cada vez que se utilice correctamente una promoción de este tipo, se agregará una tupla 'Promotion'.
- 'promotion_attributes': Contiene una cadena de caracteres con los atributos que son necesarios que contenga los metadata de la empresa utilizadora de la promoción.
 - Si 'promotion_type' es 'coupon': Se agregan los atributos mandatorios 'coupon_code' y 'total'.
 - Si 'promotion_type' es 'discount': Se agregan los atributos mandatorios 'transaction_id' y 'total'.

Se decide que el atributo 'total' sea mandatorio, ya que el mismo es necesario para poder calcular el reporte de totales para una promoción.

- 'conditions': Es una cadena de caracteres que contiene las condiciones que los 'promotion_attributes' deben tener para que se aplique el descuento.
 - Si 'promotion_type' es 'coupon': No se guarda 'IF valid_coupon_code' ya que es mandatorio. Si no se agrega ninguna condición, quedará por defecto 'true'.
 - Si 'promotion_type' es 'discount': No se guarda 'IF valid_transaction' ya que es mandatorio. Si no se agrega ninguna condición, quedará por defecto 'true'.
- 'organization': Representa la Organización propietaria de la promoción.
- 'deleted_at': Columna generada por la gema 'acts_as_paranoid', que introduce la fecha en la cual es borrada lógicamente la promoción. En caso de que no sea borrada, el campo se visualiza como 'nil'.

Promotion:

Se crea el modelo 'Promotion' para registrar cada instancia de una promoción en el sistema. Si la promoción es un cupón, se ingresarán previamente al uso. Si la promoción es un descuento, se ingresa luego del uso del descuento.

Es identificada por un 'id', 'promotion_definition_id', 'code', ya que una mismo código no puede ser repetido para una misma promoción de una organización.

Contiene los siguientes datos:

- 'id': Autogenerado internamente, es único.
- 'promotion_definition_id': Referencia al 'PromotionDefinition' al cual pertenece.
- 'code': Es el código que nombra la promoción, el mismo será:
 - Si 'PromotionDefinition' tiene como 'promotion_type' el valor 'coupon': Representará el 'coupon_code' del cupón.
 - Si 'PromotionDefinition' tiene como 'promotion_type' el valor 'discount': Representará el 'transaction_id' de la transacción.
- 'used': Boolean que determina si la 'promotion' fue usada o no.
 - Si 'PromotionDefinition' tiene como 'promotion_type' el valor 'coupon': Cuando el cupón todavía no fue utilizado, el valor será 'false', cuando el cupón fué utilizado, el valor será 'true' y no se podrá volver a utilizar dicho cupón.
 - Si 'PromotionDefinition' tiene como 'promotion_type' el valor 'discount': Representa que la transacción fue utilizada, siempre los discount, tendrán como valor 'true' el campo, ya que no se registra la transacción en la tabla, hasta que la misma sea aplicada.

4.5.5. RF5: Listado de promociones

Pre-requisitos:

- Previamente a realizar esta acción, el usuario debe haber realizado 'sign in' ([RF2: Autenticación de usuario](#))

Gemas utilizadas:

'has_scope': Provee un mecanismo sencillo para determinar scopes para filtrar datos en búsquedas.

'acts_as_tenant': Provee capacidad de 'Multi-Tenancy' a los modelos indicados.

PromotionDefinition:

Se agregó a 'PromotionDefinition' las siguientes características:

- `scope :by_state, ->(state) { where(promotion_state: state) }`
- `scope :by_type, ->(type) { where(promotion_type: type) }`
- `scope :by_name, ->(name) { where('name like ?', "%#{name}%") }`
- `scope :by_code, ->(code) { where('code like ?', "%#{code}%") }`

De esa forma, se puede conseguir el filtrado necesario para el listado de promociones.

El modelo de los 'PromotionDefinition' ya estaba hecho desde el [RF4: Gestión de promociones](#), Se listó en la vista, las promociones de la organización del cliente.

Se aseguró que el cliente sólo pudiera ver las promociones de su organización porque en el [RF1: Registro de usuario](#) se agregó la capacidad de 'Multi-Tenancy' donde cada cliente tiene como 'tenant' su organización.

4.5.6. RF6: Baja de promoción

Pre-requisitos:

- Previamente a realizar esta acción, el usuario debe haber realizado 'sign in' ([RF2: Autenticación de usuario](#))
- El usuario debe ser administrador de la organización.
- Al menos debe haber una promoción registrada en la organización ([RF4: Gestión de promociones](#))

Gemas utilizadas:

'acts_as_paranoid': Provee borrado lógico de manera sencilla.

PromotionDefinition:

Se agrega el campo 'deleted_at' que contendrá la fecha en la cual se realizó el borrado lógico. Si no tiene fecha, significa que no está borrado.

4.5.7. RF7: Reporte de uso

Pre-requisitos:

- Para usuarios que desean acceder internamente desde el sistema, previamente a realizar esta acción, el usuario debe haber realizado 'sign in' ([RF2: Autenticación de usuario](#))
- Para entidades que desean acceder externamente, tienen que incluir una 'Application Key' con los permisos necesarios ([RF3: Gestión de clave de aplicación](#)).

PromotionReport:

Se crea este modelo para guardar una tupla por cada vez que se evaluó una promoción.

- 'id': Autogenerado internamente, es único.
- 'promotion_definition_id': Referencia al 'PromotionDefinition' al cual pertenece.
- 'Response_time': Guarda la latencia entre que se recibió la petición de evaluación hasta que se envió la respuesta.
- 'spent_amount': Total del importe que se le descontó al cliente por la aplicación del descuento.
- 'Successful': 'Boolean' que indica si se evaluó como válida o no.

4.5.8. RF8: Evaluación de promoción

Pre-requisitos:

- Para entidades que desean acceder externamente, tienen que incluir una 'Application Key' con los permisos necesarios ([RF3: Gestión de clave de aplicación](#)).

Gemas utilizadas:

'jwt': Provee mecanismos para gestionar 'JWT's.

'dentaku': Parsea y evalúa fórmulas lógicas.

Ocurrida una evaluación, el sistema registra un 'PromotionReport' con los valores correspondientes y luego es guardado como una tupla en la base de datos. Estos datos, serán los que se utilizarán en el [RF7: Reporte de uso](#)

El ente externo envía un request, conteniendo los siguientes datos:

- En el 'Header':
 - 'Authentication' con el 'Application Key' correspondiente.
- En el 'Body':
 - 'code' que representa el 'PromotionDefinition' que se desea evaluar.
 - 'metadata': que contiene los atributos que son necesarios para evaluar la promoción.
 - Debe tener 'total' que representa el total de la compra.
 - Es obligatorio tener 'coupon_code' en caso de que la promoción sea del tipo cupón.
 - Es obligatorio tener 'transaction_id' en caso de que la promoción sea del tipo descuento.

El sistema reemplaza los campos que aparecen en 'promotion_attributes' de la 'PromotionDefinition' vinculada al cupón o descuento que hay que tratar y luego utiliza 'dentaku' para evaluar la expresión 'boolean' y así determinar si es válida.

5. Proceso de desarrollo, testing y deployment

5.1. Instrucciones para el desarrollo:

- Download repository

```
git clone https://github.com/ArqSoftPractica/ASP_Obl.git
```

- Gemfile dependencies

```
bundle install
```

- Creating the database

```
rake db:create
```

- Running Migrations

```
rake db:migrate
```

- Download and run Docker Container of Loggly

```
sudo docker run -d -p 514/udp --name loggly-docker -e TOKEN=TOKEN -e TAG=Docker  
sendgridlabs/loggly-docker
```

Reemplazar TOKEN por el token de customer de Loggly para tu aplicación.

- Adding config of Logglir to config/environments/.rb

```
require 'logglir'  
config.logger = Logglir.new("https://logs-01.loggly.com/inputs/TOKEN/tag/ruby/",  
:threaded => true)
```

Reemplazar TOKEN por el token de customer de Loggly para tu aplicación.

5.2. Instrucciones para las pruebas:

Pruebas automáticas:

```
rake test
```

Pruebas de carga:

- Abrir JMeter
- Ir a Archivo > Open y seleccionar el archivo 'Plan.jmx'
- Apretar el botón 'Play' en JMeter para correr las pruebas.

5.3. Instrucciones para el deploy:

- Descargar el repositorio

```
git clone https://github.com/ArqSoftPractica/ASP_Obl.git
```

- Instalar las dependencias de Gemfile

```
bundle install
```

- Crear una nueva app en Heroku, llamada 'app-coupons'
- Login en Heroku via terminal

```
heroku login
```

- Crear un nuevo repositorio Heroku para la aplicación

```
heroku git:remote -a app-coupons
```

- Configurando el 'buildpack' por defecto de la aplicación

```
heroku buildpacks:set heroku/ruby
```

- Haciendo 'Push' al repositorio Heroku.

```
git push heroku master
```

- Agregando las variables de ambiente en Heroku

```
DATABASE_URL = ...  
LANG = en_US.UTF-8  
MAIL_PASSWORD = ...  
MAIL_ADDRESS = ...  
RACK_ENV = production  
RAILS_ENV = production  
RAILS_LOG_TO_STDOUT = enabled  
RAILS_SERVE_STATIC_FILES = enabled  
SECRET_KEY_BASE = ...
```

- Correr las migraciones

```
heroku run db:migrate
```

- Configurar Loggly en Heroku

```
Heroku drains:add http://logs-01.loggly.com/bulk/TOKEN/tag/heroku --app  
HEROKU_APP_NAME
```

Reemplazar TOKEN por el token de customer de Loggly para tu aplicación.

Reemplazar HEROKU_APP_NAME por el nombre de la aplicación Heroku.

5.4. Gestionar la app en producción:

<code>heroku apps</code>	<code>(list the deployed apps)</code>
<code>heroku apps:info</code>	<code>(details of an app)</code>
<code>heroku -h</code>	<code>(help)</code>
<code>heroku logs --tail</code>	<code>(logs)</code>
<code>heroku run rake db:migrate</code>	<code>(apply migrations)</code>
<code>heroku ps</code>	<code>(process status)</code>
<code>heroku open</code>	<code>(start the app)</code>
<code>heroku run rails console</code>	<code>(run rails console)</code>
<code>heroku run bash</code>	<code>(start server console)</code>
<code>heroku git:remote -a app-coupons</code>	<code>(add remote for Git)</code>

<code>git push heroku develop:master</code>	<code>(ramaOrig:ramaDest)</code>
---	----------------------------------

Disponibilidad del servicio:

Al utilizar Heroku, se asegura la disponibilidad del servicio, pudiendo realizar actualizaciones del software en tiempo real, sin afectar la disponibilidad del servicio de producción.

6. Cumplimiento de The Twelve-Factor App

Se sigue como guía la publicación oficial: [The Twelve-Factor App](#)

1. Código base

Descripción: Un código base sobre el que hacer el control de versiones y múltiples despliegues

Como parte del [RNF6: Código fuente](#), era solicitado que se realizara un control de versiones y múltiples despliegues.

2. Dependencias

Descripción: Declarar y aislar explícitamente las dependencias

Como parte del [RNF6: Código fuente](#), era solicitado que se realizara el desarrollo del código en Ruby, y por defecto, 'Ruby' proporciona un 'GemFile' para gestionar explícitamente las dependencias del proyecto.

3. Configuraciones

Descripción: Guardar la configuración en el entorno

El [RNF3: Configuración y manejo de secretos](#), solicita que las variables de entorno sean guardadas como variables de entorno. En el 'deploy' a 'Heroku', se utilizaron las variables de entorno de 'Heroku' para poder guardar las configuraciones sensibles.

4. Backing services

Descripción: Tratar a los "backing services" como recursos conectables

Utilizamos los siguientes 'backing services':

- Loggly:
 - `https://logs-01.loggly.com/inputs/TOKEN/tag/ruby/`
- PostgreSQL:
 - `postgres://sqfkaphikslcfy:fcdd2f66bcf8099736d10b572146696b8a9746f4e8a2d93167c10a5ffaf1fc08@ec2-107-22-222-161.compute-1.amazonaws.com:5432/d4lu9u3j6sgdj6`
- Redis:
 - `redis://redistogo:1e83b1a98fee5637792a59ff19251f1b@pearlfish.redistogo.com:9530/`

5. Construir, desplegar, ejecutar

Descripción: Separar completamente la etapa de construcción de la etapa de ejecución

Como parte del [RNF6: Código fuente](#), fue solicitado que se realizara el desarrollo utilizando el framework 'Ruby on Rails', y por defecto, 'Ruby on Rails' proporciona un ambiente de manejo de 'environments' tanto para producción, como para desarrollo y test. Pudiendo así, realizar deploys en diferentes ambientes.

6. Procesos

Descripción: Ejecutar la aplicación como uno o más procesos sin estado

En el [RF1: Autenticación de usuario](#), una vez el usuario ingresa, se utilizan sesiones a nivel de navegador (cookies) para evitar los 'sticky sessions'.

A nivel de proceso, no se guarda información en memoria ram que no vaya a ser usada por el mismo proceso.

7. Asignación de puertos

Descripción: Publicar servicios mediante asignación de puertos

Nuestro 'SaaS' está vinculado a diferentes puertos dependiendo el entorno en el que se está trabajando:

- En development, el puerto en cuestión para acceder al servicio es 3000.
- En producción, el puerto en cuestión para acceder al servicio es 443.

8. Concurrencia

Descripción: Escalar mediante el modelo de procesos

En nuestro caso, no fue aplicado.

9. Desechabilidad

Descripción: Hacer el sistema más robusto intentando conseguir inicios rápidos y finalizaciones seguras

En nuestro caso, no fue aplicado.

10. Paridad en desarrollo y producción

Descripción: Mantener desarrollo, preproducción y producción tan parecidos como sea posible

- Se utilizaron los mismos servicios en producción que en desarrollo, para facilitar el deploy periódico de la aplicación.
- Se realizaron múltiples deploys manuales a lo largo del proceso (15+) haciendo que la paridad sea elevada entre desarrollo y producción.
- Se fue ajustando cada una de las librerías para que se adaptaran correctamente a ambos entornos.

11. Historiales

Descripción: Tratar los historiales como una transmisión de eventos

Como parte del [RNF8: Identificación de fallas](#), se solicitó que exista un registro de logs centralizado de al menos 24hrs de antigüedad. Nosotros decidimos realizarlo también respetando los parámetros de Twelve-Factor App. Se utiliza 'stdout' para generar salidas por defecto de los eventos de la aplicación y las mismas son publicadas directamente en 'Loggly'.

12. Administración de procesos

Descripción: Ejecutar las tareas de gestión/administración como procesos que solo se ejecutan una vez

Con la finalidad de que los procesos se ejecuten una sola vez al hacer el deploy, los procesos que hay que ejecutar para realizar los deploy en los diferentes ambientes se encuentran especificados en el 'README.md' del repositorio.

Junto con release, se adjunta en el archivo 'README.md' las instrucciones para hacer 'deploy' de la misma.

7. Pruebas

7.1. Pruebas automáticas

Se realizaron pruebas automáticas para las principales funcionalidades del sistema:

Organization_test:

Test	Resultado
with valid attributes validation succeeds	Ok
with no name validations fails	Ok
with no admin validation fails	Ok

user_test:

Test	Resultado
with valid attributes validation succeeds	Ok
with empty first_name validation fails	Ok

with a number in first_name validation fails	Ok
with more than 20 characters in first_name validation fails	Ok
with empty last_name validation fails	Ok
with a number in last_name validation fails	Ok
with more than 20 characters in last_name validation fails	Ok
with invalid email validation fails	Ok
with valid email validation succeeds	Ok
with less than 6 characters password validation fails	Ok
with 6 characters password validation succeeds	Ok
with no organization validation fails	Ok

Promotion_test:

Test	Resultado
with valid attributes validation succeeds	Ok
with atleast 1 character code validation succeeds	Ok
with true used validation succeeds	Ok
with false used validation succeeds	Ok
with nil promotion_definition validation fails	Ok

Promotion_definition_test:

Test	Resultado
with valid attributes validation succeeds	Ok
with percentage discount_type validation succeeds	Ok
with value discount_type validation succeeds	Ok
with blank discount_type validation fails	Ok
with invalid discount_type validation fails	Ok
with value of less than 0 for percentage validation fails	Ok

with value of 0 for percentage validation succeeds	Ok
with value of 100 for percentage validation succeeds	Ok
with value of more than 100 for percentage validation fails	Ok
with value of less than 0 for value validation fails	Ok
with value of 0 for value validation succeeds	Ok
with value of 100 for value validation succeeds	Ok
with value of more than 100 for value validation succeeds	Ok
with promotion_state active validation succeeds	Ok
with promotion_state inactive validation succeeds	Ok
with promotion_state blank validation fails	Ok
with promotion_state different than active or inactive validation fails	Ok
with promotion_type coupon validation succeeds	Ok
with promotion_type discount validation succeeds	Ok
with promotion_type blank validation fails	Ok
with promotion_attributes blank validation succeeds	Ok
with promotion_attributes valid validation succeeds	Ok
with conditions blank validation succeeds	Ok
with conditions valid validation succeeds	Ok
returns the promotion definitions filtered by the given state	Ok
returns the promotion definitions filtered by the given type	Ok
returns the promotion definitions filtered by the given name	Ok
returns the promotion definitions filtered by the given code	Ok

Promotion_report_test:

Test	Resultado
with valid attributes validation succeeds	Ok

with no response time validation fails	Ok
with no spent_amount validation fails	Ok
with nil promotion_definition validation fails	Ok
returns the reports related to the given promotion_definition	Ok
returns the reports with successful calls	Ok

7.2. Pruebas de carga

Con 'throughput' constante:

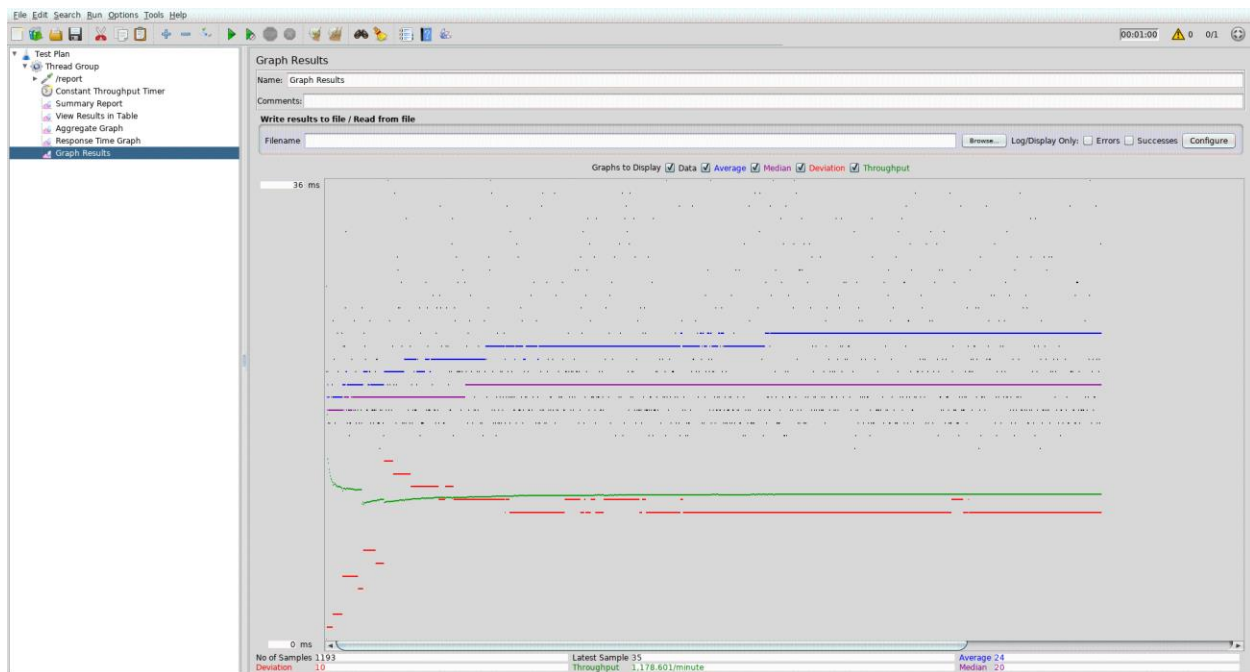
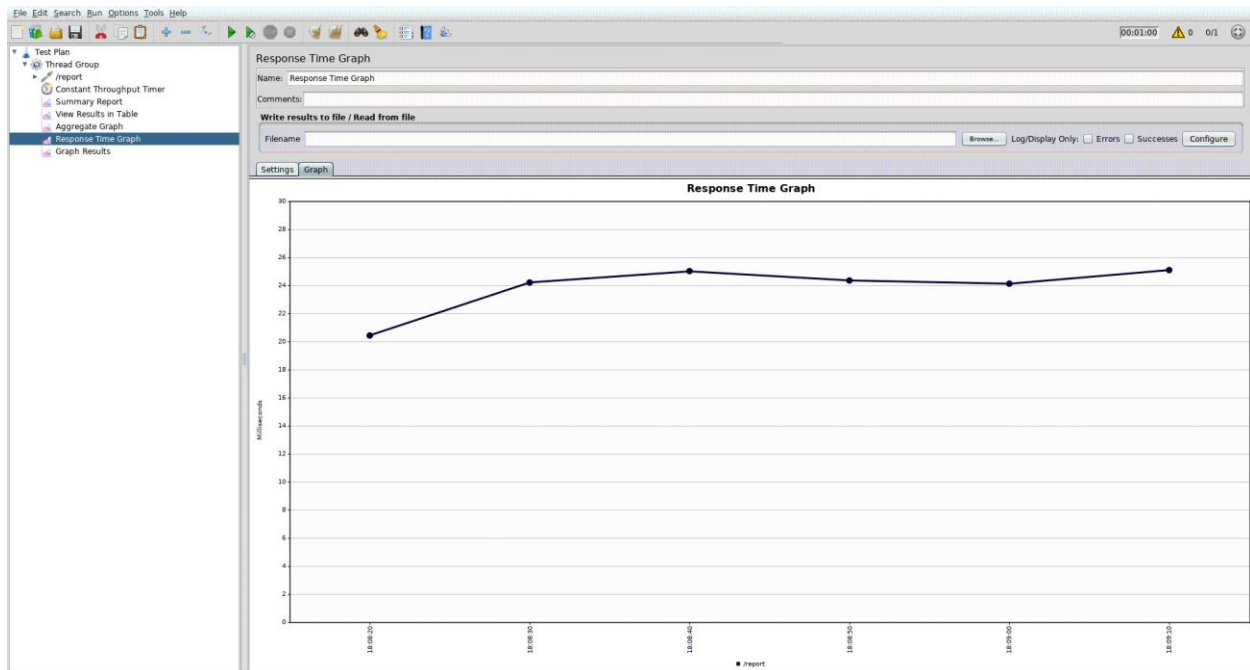
The screenshot shows the JMeter Summary Report window. The left sidebar lists the test plan components: Test Plan, Thread Group, /report, Constant Throughput Timer, Summary Report, View Results in Table, Aggregate Graph, Response Time Graph, and Graph Results. The main window displays the Summary Report for the 'Constant Throughput Timer'.

Summary Report
Name: Summary Report
Comments:

Write results to file / Read from file
Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes ☐ Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K/sec	Sent K/sec	Avg. Bytes
/report	1193	24	15	148	10.78	0.08%	19.6/sec	11.85	5.62	617.7
TOTAL	1193	24	15	148	10.78	0.08%	19.6/sec	11.85	5.62	617.7

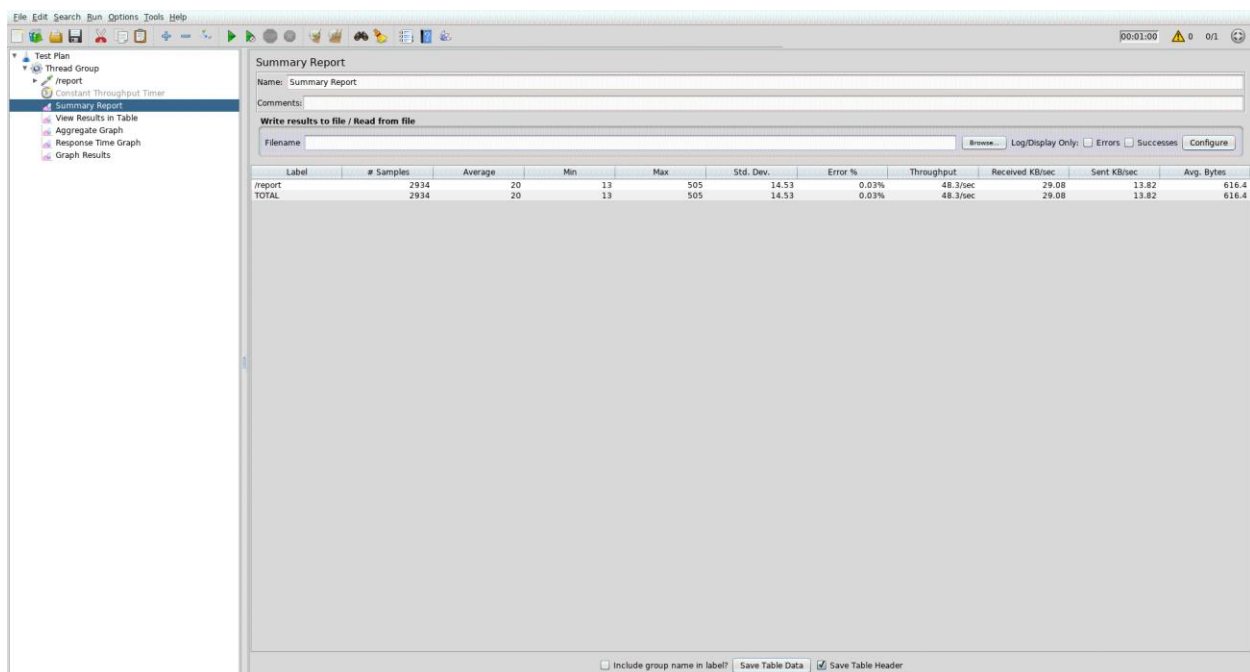
☐ Include group name in label? ☐ Save Table Data ☒ Save Table Header



Resultados:

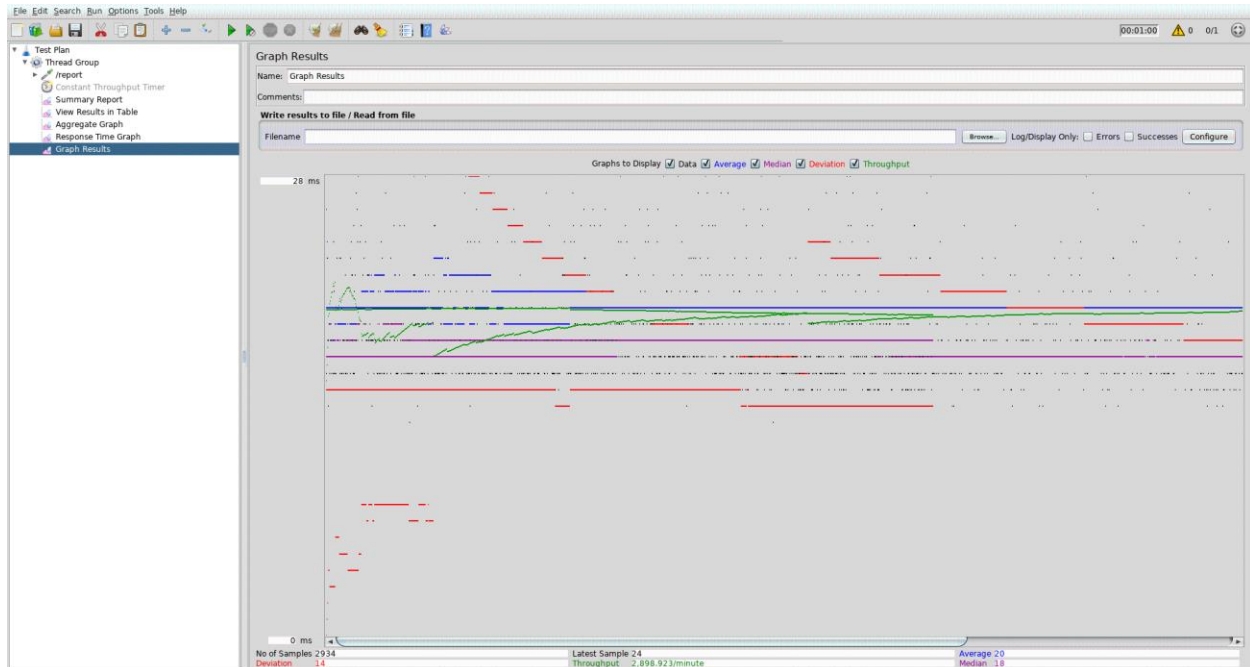
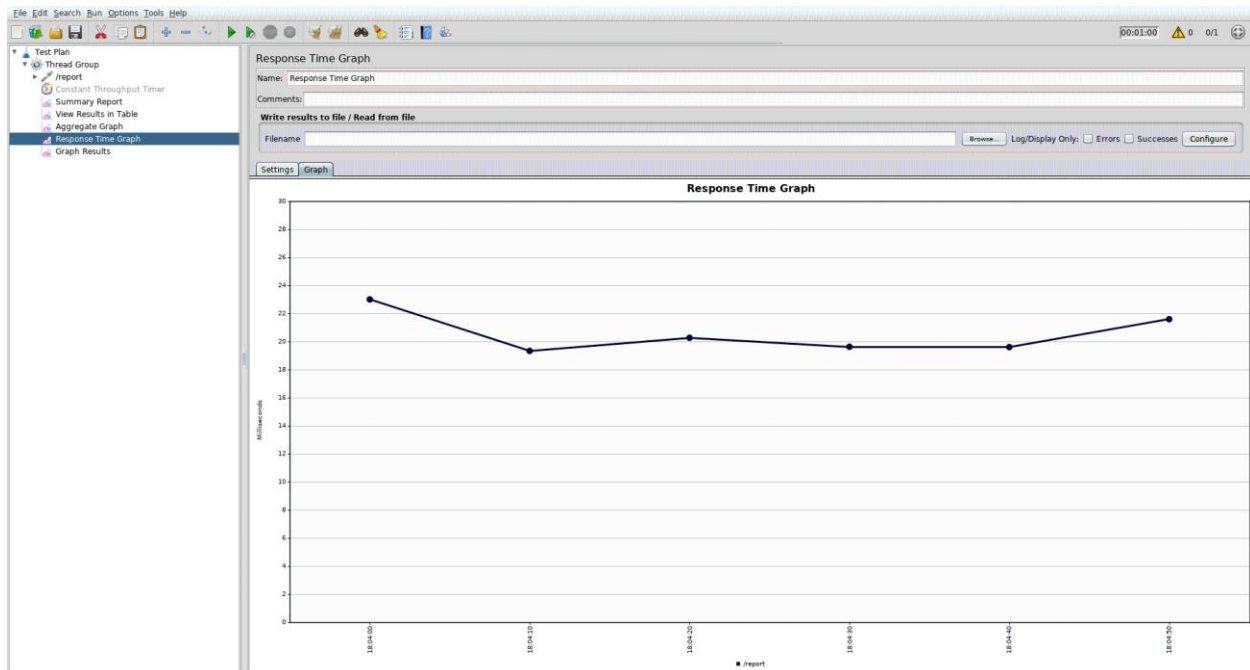
- Número de samples: 1193
- Average: 24
- Median: 20
- Deviation: 10
- Throughput: 1.178.607/min

Con 'throughput' no constante:



The screenshot shows a software interface for a test plan summary report. The window has a menu bar (File, Edit, Search, Run, Options, Tools, Help) and a toolbar. On the left is a tree view of the test plan structure, including 'Test Plan', 'Thread Group', 'report', 'Constant Throughput Timer', 'Summary Report', 'View Results in Table', 'Aggregate Graph', 'Response Time Graph', and 'Graph Results'. The main area displays the 'Summary Report' for the 'report' group. It includes fields for 'Name' (Summary Report) and 'Comments'. Below these is a section 'Write results to file / Read from file' with a 'Filename' field and buttons for 'Browse...', 'Log/Display Only', 'Errors', 'Successes', and 'Configure'. A table of results is shown with columns: Label, # Samples, Average, Min, Max, Std. Dev., Error %, Throughput, Received KB/sec, Sent KB/sec, and Avg. Bytes. The table has two rows: 'report' and 'TOTAL'. At the bottom, there are checkboxes for 'Include group name in label?', 'Save Table Data', and 'Save Table Header'.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
/report	2934	20	13	505	14.53	0.03%	48.3/sec	29.08	13.82	616.4
TOTAL	2934	20	13	505	14.53	0.03%	48.3/sec	29.08	13.82	616.4



Resultados:

- Número de samples: 2934

- Average: 20
- Median 18
- Deviation: 14
- Throughput: 2.898.923/min

8. Gestión de proyecto

8.1. Metodología

Se utilizó metodología ágil (adaptada a nuestro caso), para el desarrollo del proyecto.

Se realizó un tablero Kanban, el cual tenía las siguientes características:

- Por defecto, las 'cards' aparecen en la columna 'To do'.
- Cuando un integrante del equipo decide realizar una 'card', la pasa a la columna 'In progress'.
 - Los 'commits' realizados durante la implementación de la card, son realizados en un nuevo 'branch' representativo.
- Cuando una 'card' que está en 'In progress' es finalizada, y se verifica que cumple con los criterios de aceptación; se procede a generar un 'Pull Request', el cual posterior a una revisión, se 'mergea' en develop. Este proceso, hace que la 'card' vaya a la columna 'Done'.

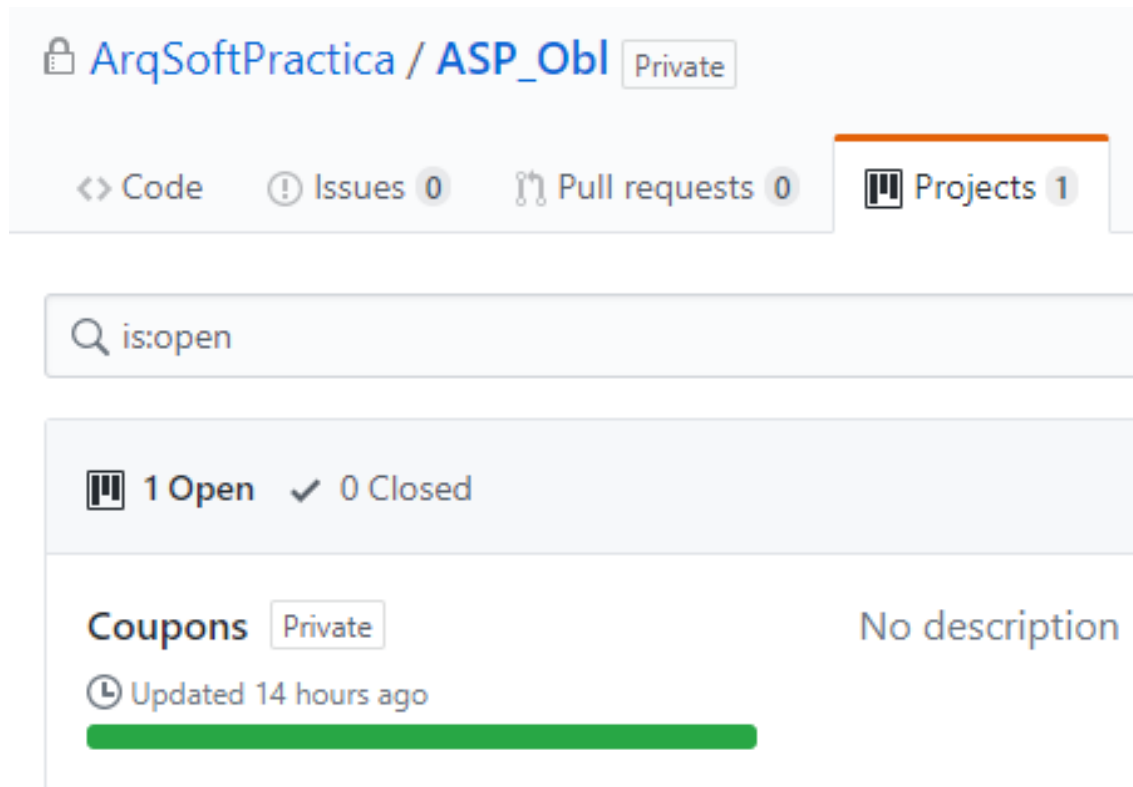
8.2. Herramientas

Se utilizó la sección de 'Projects' de Github para realizar toda la gestión de proyectos.

Cada 'card' era un 'issue' y se utilizó un tablero Kanban por defecto que proporciona Github.

Se puede ver más detalles ingresando al [tablero del proyecto](#).

Proyecto:



Tablero:

Coupons
Updated 14 hours ago

0 To do

0 In progress

18 Done

RNF 1. Performance

1 of 1

#11 opened by matiashrmdz

RNF 7. Tests

2 of 2

#17 opened by matiashrmdz

RNF 6. Source Code

7 of 7

#16 opened by matiashrmdz

RF7. Promotions Reports

7 of 7

Formato de una card de un RNF:

RNF 2. Reliability and Availability #12

Closed matiashrmdz opened this issue 22 days ago · 1 comment



matiashrmdz commented 22 days ago · edited

+ 😊 ...

- ☒ **Expose a Health Endpoint**
Expose a GET /healthcheck endpoint that respond 200 OK when every check is OK. If not, an error should be send.
- ☒ **Check databases connection**
When called GET /healthcheck, check database connectivity and respond 200 OK when the database connectivity is OK. If not, respond an error 'There is a connectivity problem with the database'.
- ☒ **Check queues connection**
When called GET /healthcheck, check queues connectivity and respond 200 OK when the availability is OK. If not, respond an error 'There is a connectivity problem with the queues'.
- ☒ **Check end points connection**
When called GET /healthcheck, check availability to receive requests and respond 200 OK when the availability is OK. If not, respond an error 'There is a connectivity problem with the endpoints'.

Formato de una card de un RF:

RF3. Manage Application Key #5



matiasrmdnz opened this issue 25 days ago · 1 comment



matiasrmdnz commented 25 days ago · edited ▼



Card

As an admin of an organization

I want to generate application keys identified by name and with the promotions associated with it
So that other organization can invoke services with this key and access the promotions in it.

Acceptance Criteria

☒ Scenario: Generate Application Key

Given an Admin of an organization is signed in

And the name selected is valid

And the name selected is not already in use by another Application Key in the organization

And there is atleast 1 promotion associated

And the promotions associated are valid

When the Admin hits the Generate button

Then the systems generates an Application Key with those attributes selected.

☒ Scenario: Generate Application Key with no promotions associated

Given an Admin of an organization is signed in

And there is no promotion associated

When the Admin hits the Generate button

Then the systems generates an Application Key with no attributes.

8.3. Gestión del Alcance

El alcance del proyecto abarca los requerimientos solicitados en el obligatorio, hasta la entrega del producto una vez finalizado.

Release - 10 de Octubre del 2019	
Card	Criterios de aceptación
RNF1. Performance	1 de 1
RNF2. Reliability and Availability	4 de 4
RNF3. Configuration and secret's handling	1 de 1
RNF4. Authentication and Authorization	4 de 4
RNF5. Security	1 de 1
RNF6. Source Code	7 de 7
RNF7. Tests	2 de 2

<i>RF8. Identify faults</i>	2 de 2
<i>RF1. Signup via Invitation Link</i>	6 de 6
<i>RF1. Signup via Web</i>	6 de 6
<i>RF2. Signin</i>	4 de 4
<i>RF3. Managing Application Key</i>	2 de 2
<i>RF4. Register Promotion</i>	1 de 1
<i>RF5. Show Promotions List</i>	6 de 6
<i>RF6. Deregister Promotion</i>	1 de 1
<i>RF7. Promotions Reports</i>	7 de 7
<i>RF8. Promotion Evaluation</i>	4 de 4

9. Detalle de Endpoints

La aplicación cuenta con dos endpoints públicos sobre los que se ha implementado el control y validación de tokens JWT (JSON Web Tokens) para todos los request que se realicen sobre los mismos.

Al utilizarse JWT, en cada request es requerida la presencia en el HEADER de la KEY Authorization donde debe viajar el token que se le proveyó a la empresa oportunamente para que pudiera consumir o consultar reporte de uso de alguna de las promo que tiene habilitadas a tales efectos.

Evaluación de Promoción

/promotions/evaluation	POST	GET, PUT, DELETE
Header: Authorization: token Body:	(200) - si pudo evaluar ok retorna {“success”: “Valid promotion”}	(404) - Not Found

<pre>{ "code": "promo_code", "metadata": { "total": int, "transaction_id": "id", "attributes": { "attr1": int, "attr2": int }} }</pre> <p>Nota: transaction_id podría sustituirse por coupon_code si se trata de una promo tipo Coupon</p>	<p>(401) - Unauthorized, el request llega sin token o el token es incorrecto</p> <p>(422) - si no estaba disponible retorna {"errors": "Invalid coupon"}</p>	
---	--	--

Reporte de Uso

/promotions/:id/report	GET	POST, PUT, DELETE
Header: Authorization: token Body: null	(200) - retorna el reporte de uso de la promoción cuyo código recibe como parte de la uri (401) - Unauthorized, el request llega sin token o el token es incorrecto	(404) - Not Found

10. Descripción de la Interfaz de Usuario

A continuación presentamos las distintas vistas de la solución, las cuales fueron diseñadas intentando favorecer la usabilidad por parte de los visitantes.



Home Page: <https://app-coupons.herokuapp.com>

Este login permite ingresar las credenciales de los usuarios previamente registrados en el sistema, y de esta forma poder acceder a las distintas operaciones habilitadas según su perfil.

The screenshot shows the 'New User' registration form. At the top left, there is a 'Coupons' logo and a 'New User' tab. The form is a light orange box with the following fields: 'Organization Name' (text input), 'Profile picture' (with a 'Choose File' button and 'No file chosen' text), 'Name' (text input), 'Last Name' (text input), 'Email' (text input), 'Password' (text input), and 'Password Confirmation' (text input). At the bottom of the form are two buttons: 'Create User' (red) and '<< Back' (light orange).

Sign up: <https://app-coupons.herokuapp.com/users/new>

Esta página permite la creación de nuevos usuarios, de ser aprobada la creación, el usuario quedará registrado como Administrador de la Organización que haya ingresado en este formulario.

Nota: haciendo clic sobre la palabra Coupons en la navigation bar, se retorna siempre a la home del sitio.

The screenshot shows the 'Promotions' management page. At the top left, there is a 'Coupons' logo and a 'Promotions' tab. At the top right, there is a 'Logout' button. The page contains a filter section with inputs for 'Filter by name', 'Filter by code', and 'Filter by status' (set to 'All'), and a dropdown for 'Filter by type' (set to 'All'). There are 'Clear filters' and 'Apply filters' buttons. Below the filters is a red 'Add Promotion' button. A table lists the promotions with columns: 'Name', 'Code', 'State', 'Type', and 'Actions'. The table has one row for 'Navidad' with code 'jlt0BcYq', state 'active', and type 'coupon'. The 'Actions' column for this row contains buttons: 'View', 'Edit', 'Delete', 'Report', and 'Generate coupons'. Below the table are two red buttons: 'Create Token' and 'User Profile'.

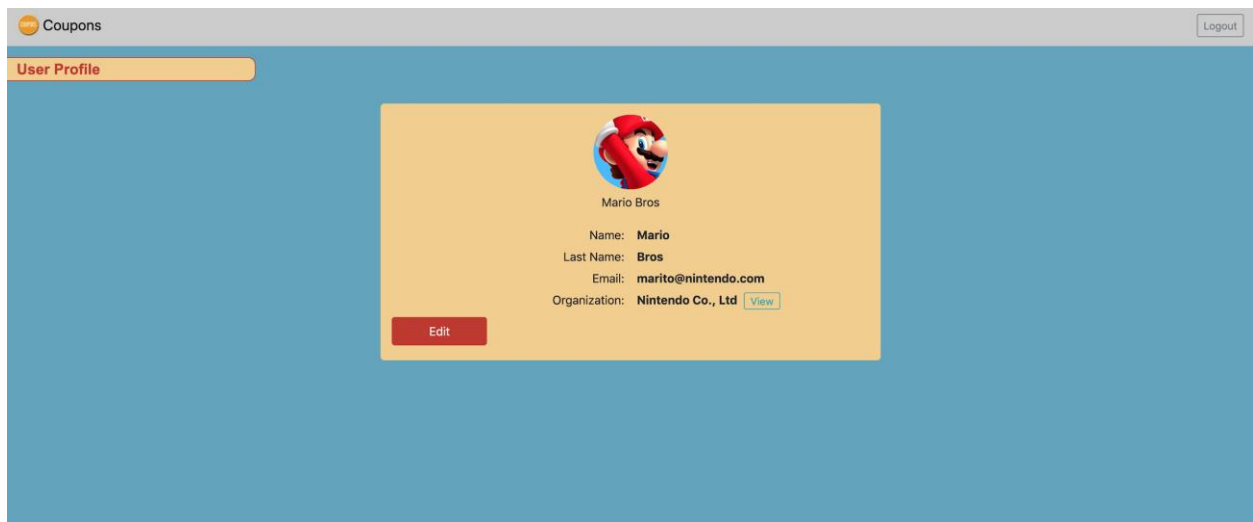
Name	Code	State	Type	Actions
Navidad	jlt0BcYq	active	coupon	View Edit Delete Report Generate coupons

Main Page: <https://app-coupons.herokuapp.com>

Esta página es la primera que se le presenta al usuario una vez validadas sus credenciales. Sobre la esquina superior derecha (dentro de la NavBar) aparece un botón Logout, el que permite finalizar la sesión y volver a la página de Login.

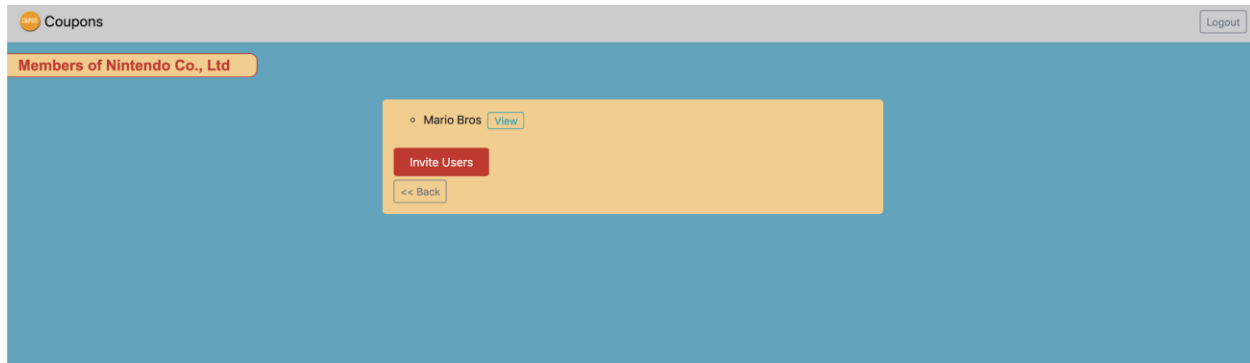
En esta página se visualizan todas las promociones que tienen definidas hasta el momento la organización (en este caso una sola, la promo Navidad), sobre las cuales se pueden realizar un conjunto de acciones (View, Edit, Delete, Report, Generate coupons). Además se puede generar una nueva clave de aplicación haciendo clic sobre el botón Generar Token. También se permite desde esta vista, visualizar los datos del usuario loggeado, para ello se debe hacer clic sobre el botón User Profile.

Sobre la parte superior de esta pantalla, se aprecian varios campos que permiten realizar un filtrado sobre las promos que se muestran en el listado, pudiendo definir distintos patrones para desplegar sólo el subconjunto requerido en cada momento.



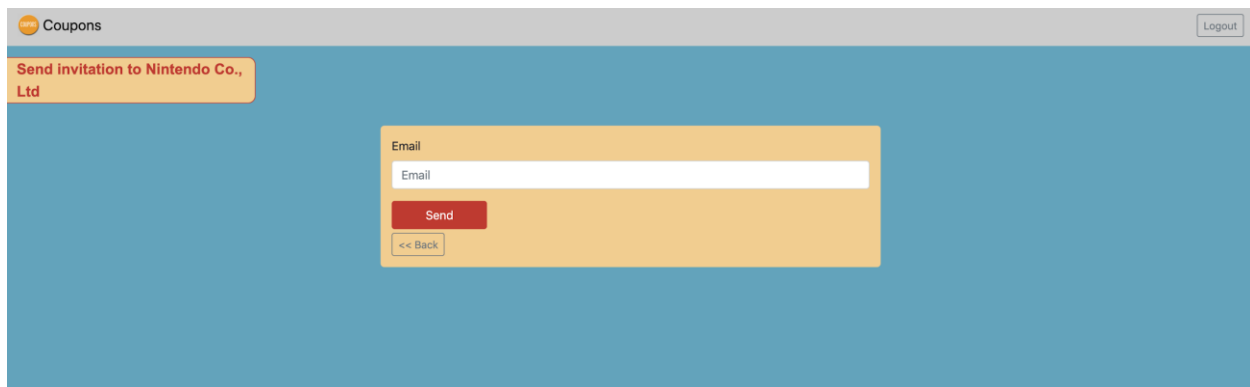
User Profile: <https://app-coupons.herokuapp.com/users/{id}>

Desde la vista User Profile, se pueden observar todos los datos del usuario y realizar modificaciones sobre los mismos haciendo clic sobre el botón Edit. Desde aquí también se puede acceder fácilmente a la vista que presenta la lista de miembros de la Organización a la que pertenece dicho usuario, para ello simplemente se debe hacer clic sobre el botón View que se encuentra a continuación del nombre la misma.



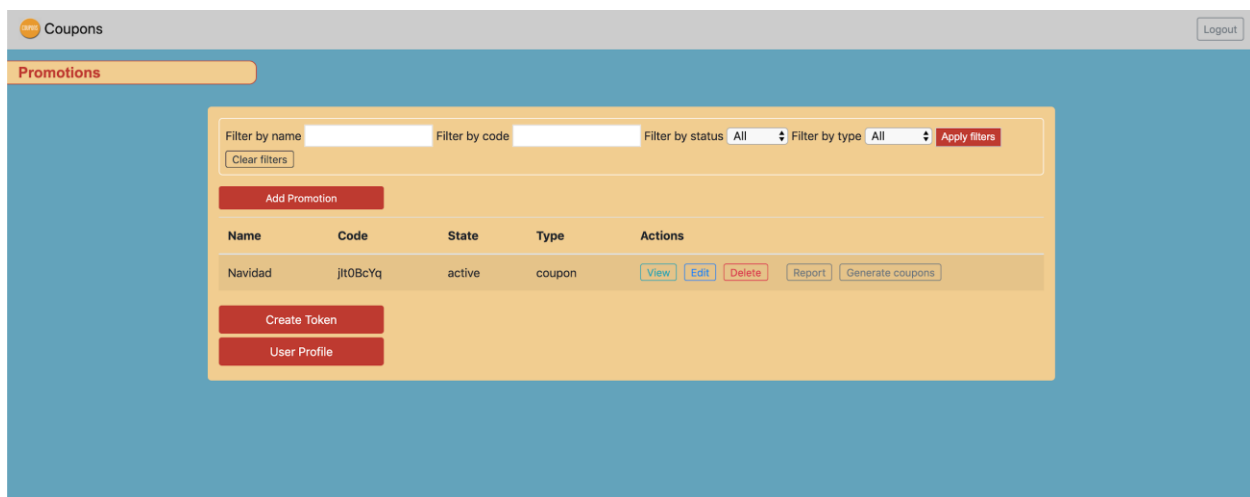
Organization members: <https://app-coupons.herokuapp.com/organizations/{id}>

En este caso la organización únicamente cuenta con un miembro, pero esto se puede solucionar invitando nuevos usuarios, lo cual, puede ser realizado desde aquí mismo dando clic al botón Invite Users.



Send invitation: <https://app-coupons.herokuapp.com/invitation/form>

Desde aquí ya se puede ingresar el mail de la persona a la que se desee invitar. Ese mail le llegará con toda la información necesaria para que pueda finalizar el registro como nuevo miembro de la organización.



Main Page: <https://app-coupons.herokuapp.com>

De vuelta en la página principal, ahora vamos a proceder a crear una nueva promoción. Para ello debemos hacer clic sobre el botón Add Promotion, que se encuentra situado entre el filtro y la lista de promociones de la organización.

The screenshot shows the 'New promotion' form in the app-coupons application. The form is titled 'New promotion' and is located on the left side of the page. It contains the following fields and options:

- Name:** A text input field with the value 'Carnaval'.
- Discount type:** Radio buttons for 'Percentage' (selected) and 'Value'.
- Promotion type:** Radio buttons for 'Coupon' (selected) and 'Discount'.
- Value:** A text input field with the value '20'.
- Attributes:** A text area for entering attributes, with a note: 'Attributes must be inserted comma separated. Default attributes 'total' and 'coupon_code' or 'transaction_id' already included.'
- Conditions:** A text input field with the value 'total > 10000'.
- Buttons:** A red 'Create' button and a '<< Back' button.

New Promotion: https://app-coupons.herokuapp.com/promotion_definitions/new

Para cada nueva promoción que se desee definir, se deben ingresar todos los datos que se muestran en la imagen anterior. Estos son: el nombre, el tipo de descuento, el tipo de promo, el valor asociado al tipo de descuento, la lista de atributos que se desee evaluar, y las condiciones a evaluar sobre los atributos para validar que la promoción puede ser utilizada.

En este caso, estamos agregando una nueva promo para Carnaval, que otorga un 20% de descuento si el total de la compra supera los \$10.000.

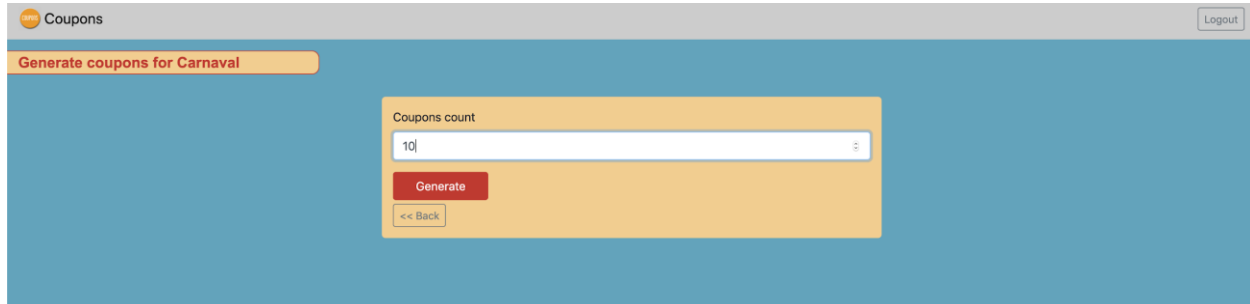
The screenshot shows the 'Promotions' page in the app-coupons application. The page has a header with 'Coupons' and a 'Logout' button. The main content area is titled 'Promotions' and contains the following elements:

- Filter by name:** A text input field.
- Filter by code:** A text input field.
- Filter by status:** A dropdown menu with 'All' selected.
- Filter by type:** A dropdown menu with 'All' selected.
- Buttons:** 'Apply filters' (red), 'Clear filters' (orange), 'Add Promotion' (red), 'Create Token' (red), and 'User Profile' (red).
- Table:** A table with columns 'Name', 'Code', 'State', 'Type', and 'Actions'. It contains two rows of data:

Name	Code	State	Type	Actions
Carnaval	rHdW63Ah	active	coupon	View Edit Delete Report Generate coupons
Navidad	jIt0BcYq	active	coupon	View Edit Delete Report Generate coupons

Main Page: <https://app-coupons.herokuapp.com>

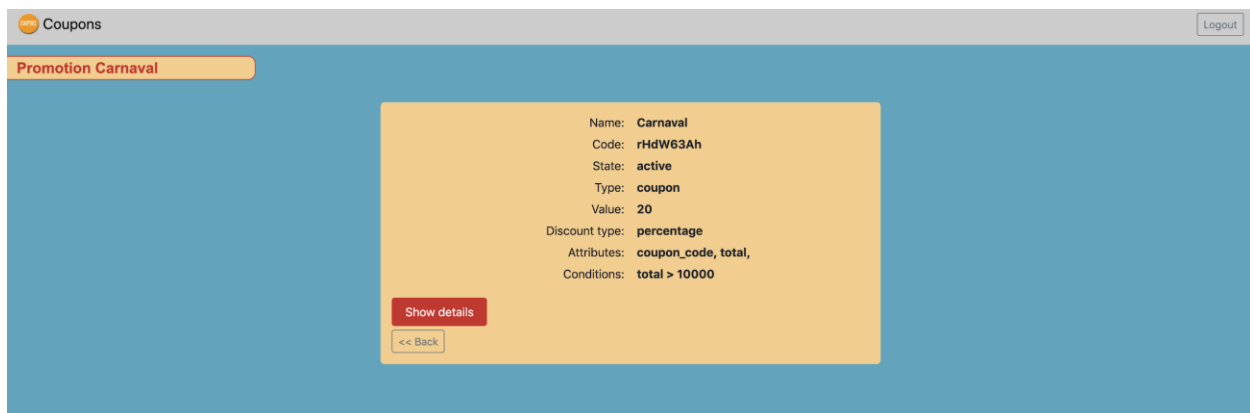
La nueva promo aparece en el listado de promociones de la página principal. Ahora creemos algunos cupones para que la promo pueda comenzar a utilizarse, para ello debemos hacer clic sobre el botón Generate coupons que se encuentra al final de la misma línea de la promo.



The screenshot shows a web application interface with a header bar labeled 'Coupons' and a 'Logout' button. Below the header, there is a section titled 'Generate coupons for Carnaval'. Inside this section, there is a form with a label 'Coupons count' and a text input field containing the number '10'. Below the input field, there are two buttons: a red 'Generate' button and a light blue '<< Back' button.

Generate coupons: https://app-coupons.herokuapp.com/promotion_definitions/{id}/promotions/new

Generamos 10 cupones para esa promo. Una vez que los mismos se terminan de generar (puede demorar unos segundos, esto se realizar en background), podemos desde la Main Page hacer clic sobre View en la misma línea de la promo y pasaremos a ver los datos de esa promoción tal cual lo muestra la siguiente imagen.



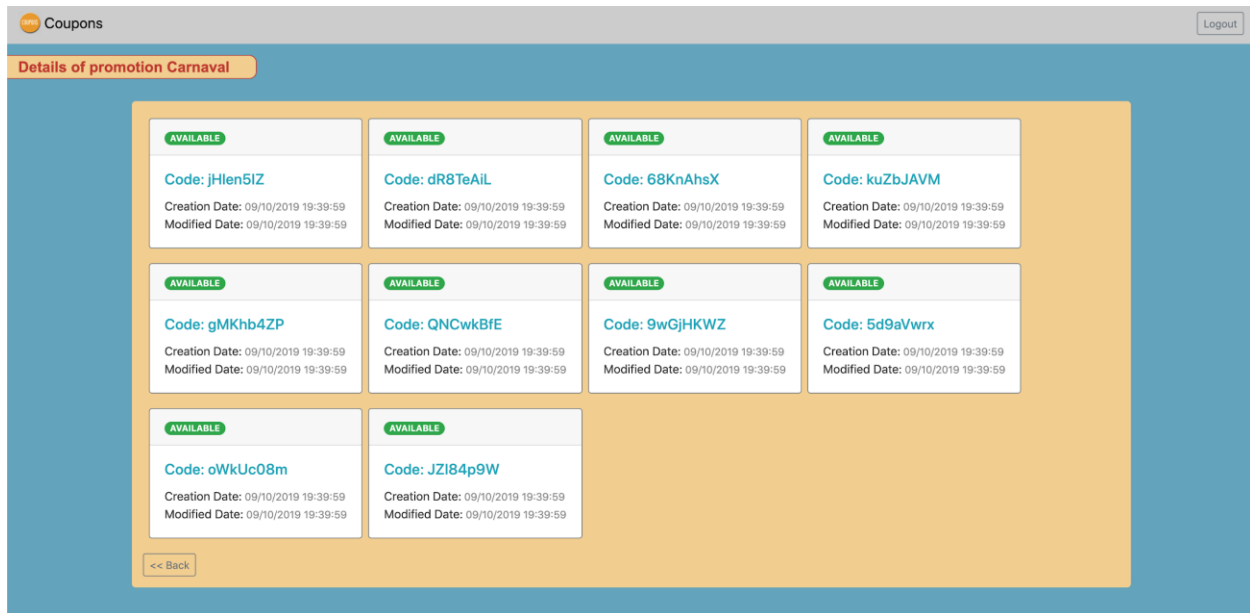
The screenshot shows a web application interface with a header bar labeled 'Coupons' and a 'Logout' button. Below the header, there is a section titled 'Promotion Carnaval'. Inside this section, there is a form displaying the details of a promotion. The details are as follows:

Name:	Carnaval
Code:	rHdW63Ah
State:	active
Type:	coupon
Value:	20
Discount type:	percentage
Attributes:	coupon_code, total,
Conditions:	total > 10000

Below the details, there are two buttons: a red 'Show details' button and a light blue '<< Back' button.

Promotion: https://app-coupons.herokuapp.com/promotion_definitions/{id}

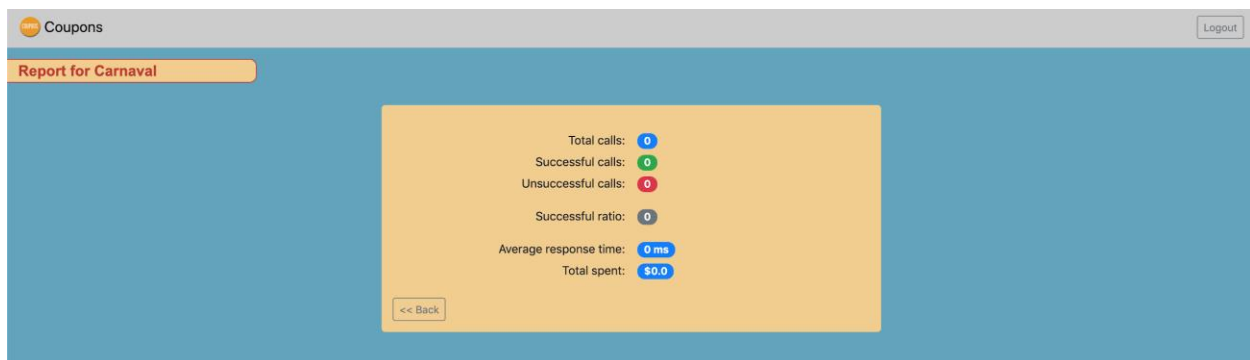
Todos los datos de la promoción se pueden apreciar fácilmente desde esta vista. Para ver los cupones que habíamos creado algunos pasos atrás, debemos hacer clic sobre el botón Show details.



Details of a promotion: https://app-coupons.herokuapp.com/promotion_definitions/{id}/promotions

En esta vista se pueden ver todos los cupones (o transacciones en caso de ser una promo de tipo descuento), en este caso para los cupones podemos ver que todos se encuentran disponibles.

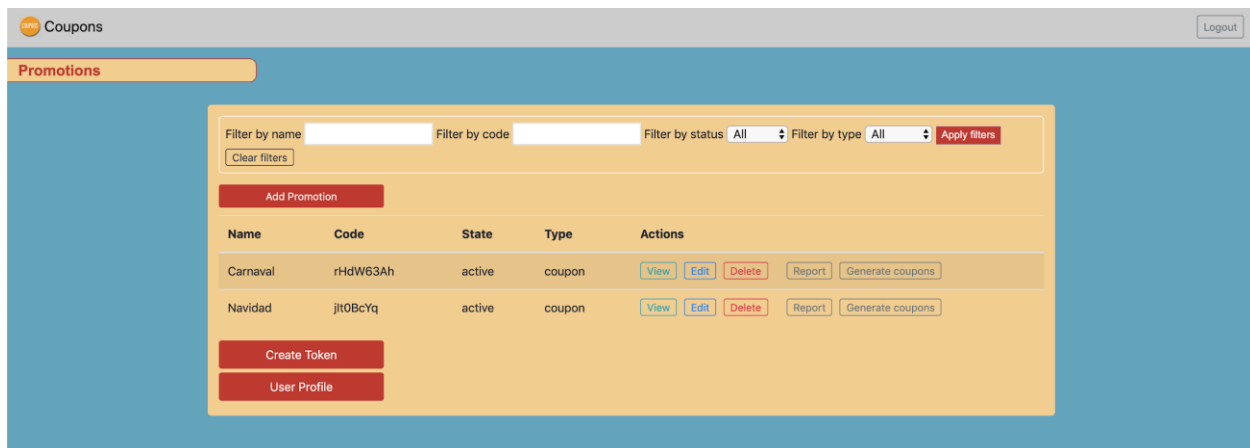
Si volvemos a la Main Page y hacemos clic sobre el botón Report en la misma línea de la promo, pasaremos a ver las estadísticas de uso de dicha promoción.



Report: https://app-coupons.herokuapp.com/promotion_definitions/{id}/report

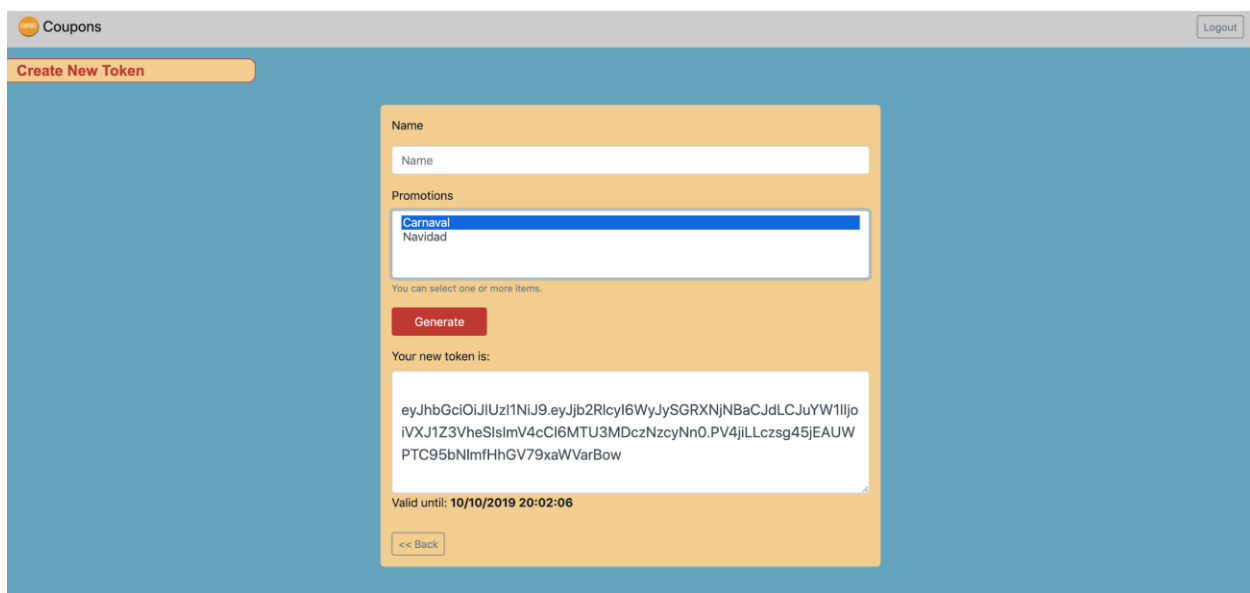
En este caso, como la promoción Carnaval es completamente nueva, y no ha sido utilizada todavía, todos los contadores del reporte se muestran en cero.

Para finalizar volvamos a la principal para generar un token y que los clientes puedan comenzar a hacer uso de la misma.



Main Page: <https://app-coupons.herokuapp.com>

Debemos hacer clic sobre el botón Generar Token, esto nos presentará la siguiente página donde podemos generar el token JWT para que los usuarios pueden consumir es (u otras) promo.



Main Page: https://app-coupons.herokuapp.com/promotions/generate_token

Para generar un nuevo token de aplicación, una vez en esta vista debemos ingresar un nombre, seleccionar el o las promos que se quieren permitir utilizar con este token, y hacer clic sobre el botón Generate. El token generado aparecerá en pantalla y puede desde ahí ser copiado y distribuido según corresponda.