

# **Universidad ORT Uruguay**

## **Facultad de Ingeniería**

### **Arquitectura de Software en la Práctica**

#### **Obligatorio 2**

Matias Hernández 169236  
Diego Klappenbach 178226  
Esteban Muzio 92391

Docente: Pablo Vilas

**2019**

# Índice

<b>Índice</b>	<b>2</b>
<b>1. Introducción</b>	<b>5</b>
<b>2. Antecedentes</b>	<b>5</b>
2.1 Propósito del sistema	5
<b>3. Descripción de la arquitectura</b>	<b>6</b>
3.2 Vista de componentes y conectores	6
3.2.1. Representación primaria	6
3.2.2. Catálogo de elementos	7
3.2.3. Interfaces	8
3.2.5. Decisiones de diseño	11
3.3 Vista de despliegue	12
3.3.1. Representación primaria	12
3.3.2. Catálogo de elementos	13
3.3.3. Relación con componentes	14
3.3.4. Decisiones de diseño	15
<b>4. Justificaciones de diseño</b>	<b>16</b>
4.1 Versiones de Ruby y Ruby on Rails	16
4.2 Entorno de desarrollo	16
4.3 Librerías de terceros (Gemas)	17
4.4. Justificación de los Requerimientos No Funcionales	20
4.4.1. RNF1: Performance	20
4.4.2. RNF2: Confiabilidad y disponibilidad	20
4.4.3. RNF3: Configuración y manejo de secretos	20
4.4.4. RNF4: Autenticación y autorización	21
4.4.5. RNF5: Seguridad	22
4.4.6. RNF6: Código fuente	23
4.4.7. RNF7: Pruebas	24
4.4.8. RNF8: Identificación de fallas	24
4.4.9. RNF9: Independencia de aplicaciones	26
4.4.10. RNF10: Monitориabilidad	30
4.5. Justificación de los Requerimientos Funcionales	31
4.5.1. RF1: Registro de usuario	31
4.5.2. RF2: Autenticación de usuario	32

4.5.3. RF3: Gestión de clave de aplicación	32
4.5.4. RF4: Gestión de promociones	33
4.5.5. RF5: Listado de promociones	35
4.5.6. RF6: Baja de promoción	36
4.5.7. RF7: Reporte de uso	36
4.5.8. RF8: Evaluación de promoción	37
4.5.9. RF9: Gestión de usuarios	38
4.5.10. RF10: SDK de evaluación de promociones	39
4.5.11. RF11: Límite de uso de cupón	41
4.5.12. RF12: Expiración de promociones	42
4.5.13. RF13: Agregar cupones a promoción	43
4.5.14. RF14: Vencimiento de cupones	44
4.5.15. RF15: Reporte demográfico de usuarios	45
<b>5. Proceso de desarrollo, testing y deployment</b>	<b>47</b>
5.1. Instrucciones para el desarrollo	47
5.1.1. app-coupons-reports	47
5.1.2. app-coupons-web	47
5.1.3. app-coupons-api	48
5.1.4. app-webproxy	49
5.1.5. app-amqp	49
5.1.6. coupons-sdk	50
5.1.7. app-coupons-client	50
5.2. Instrucciones para las pruebas	51
5.3. Instrucciones para el deploy	52
5.3.1. app-coupons-reports	52
5.3.2. app-coupons-api	53
5.3.3. app-coupons-web	55
5.3.4. app-webproxy	56
5.3.5. app-amqp	56
5.3.6. coupons-sdk	57
5.3.7. app-coupons-client	58
5.4. Gestionar la app en producción:	59
<b>6. Cumplimiento de The Twelve-Factor App</b>	<b>60</b>
<b>7. Pruebas</b>	<b>63</b>
7.1. Pruebas automáticas	63
7.1.1. app-coupons-api	63
7.1.2. app-coupons-reports	66
7.2. Pruebas de carga	68

<b>8. Gestión de proyecto</b>	<b>72</b>
8.1. Metodología	72
8.2. Herramientas	72
8.3. Gestión del Alcance	75
<b>9. Detalle de Endpoints</b>	<b>77</b>
<b>10. Descripción de la Interfaz de Usuario</b>	<b>80</b>

# 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

Luego del éxito obtenido con el despliegue de la aplicación Coupons en los primeros clientes, su empresa encuentra dificultades al escalar el producto.

A partir de un análisis detallado, descubre que existen los siguientes puntos de tensión en la arquitectura:

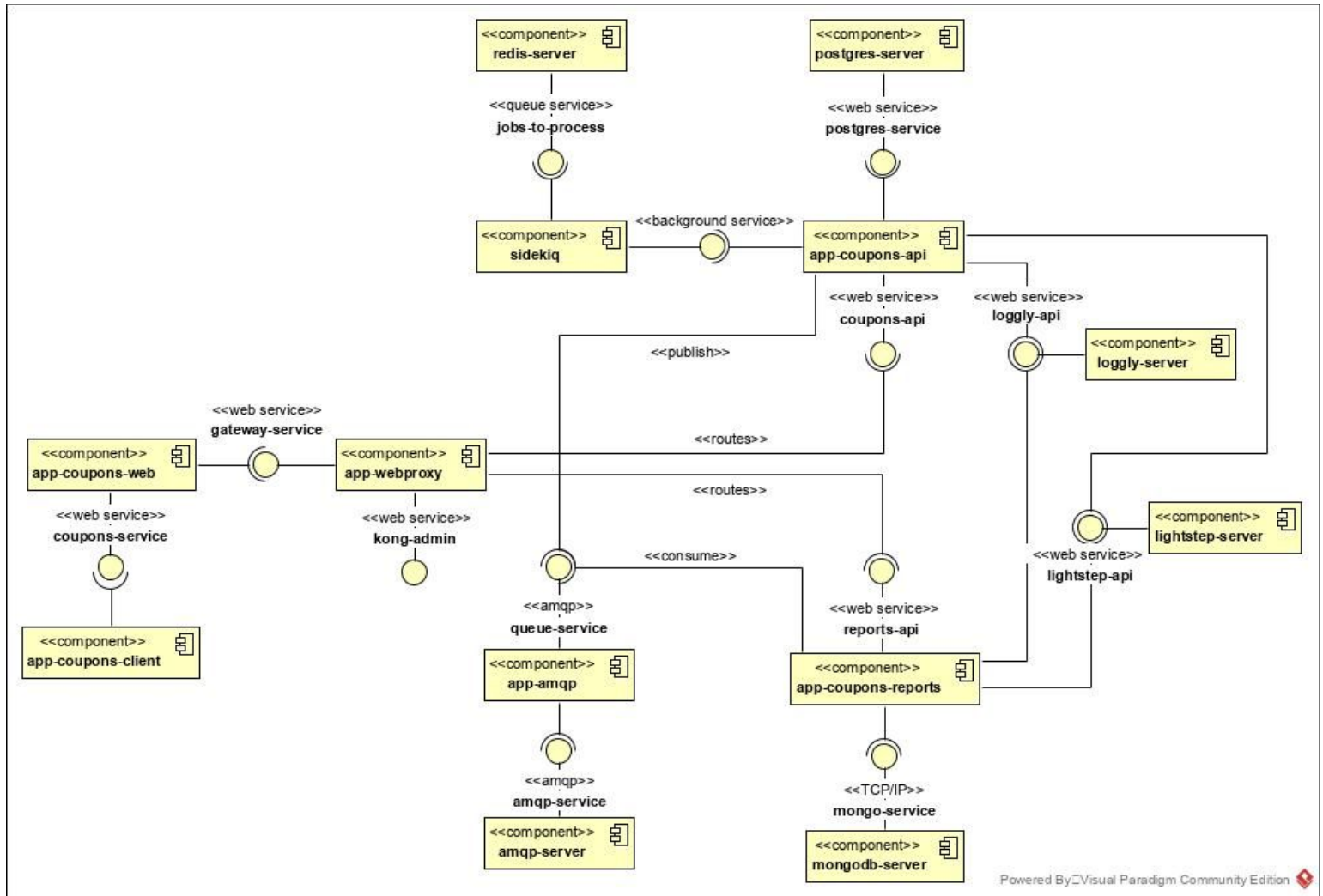
- Al acceder a los reportes la performance de evaluación de promociones y administración de promociones se ve degradada.
- En horas pico la performance global del sistema se ve perjudicada debido a la elevada cantidad de peticiones sobre el endpoint de evaluación.

Por otra parte, su empresa consiguió un cliente del Fortune 100. Si bien esto significa un gran paso para la empresa se debe realizar modificaciones para atenderse a sus requerimientos.

### 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-api</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>
<i>lightstep-server</i>	<i>Lightstep Server Api</i>	<i>Servicio de tracing, que realiza trazabilidad entre los servicios app-coupons-api y app-coupons-reports.</i>
<i>app-coupons-reports</i>	<i>Web Server Api</i>	<i>Servicio proveedor de servicios de reportería.</i>
<i>mongodb-server</i>	<i>MongoDB Server</i>	<i>Servicio web que actúa como interfaz para poder acceder a la base de datos MongoDB.</i>
<i>app-amqp</i>	<i>Web Server Api</i>	<i>Servicio proveedor de cola de mensajería.</i>
<i>amqp-server</i>	<i>RabbitMQ Server</i>	<i>Servicio que implementa la cola de mensajería utilizando RabbitMQ.</i>
<i>app-webproxy</i>	<i>Api Gateway</i>	<i>Servicio que implementa el patrón Api Gateway.</i>
<i>app-coupons-web</i>	<i>Web Server</i>	<i>Servicio que expone la interfaz al usuario.</i>
<i>app-coupons-client</i>	<i>Demo client</i>	<i>Cliente demo que implementa el SDK.</i>

### 3.2.3. Interfaces

Interfaz	<i>postgres-service</i>
La implementa	<i>postgres-server</i>
Servicio	Descripción
<i>Api para manejar PostgreSQL.</i>	<i>Es utilizado por app-coupons-api para guardar y recolectar los datos que necesitan ser persistidos de la app-coupons-api.</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>background service</i>
La implementa	<i>sidekiq</i>
Servicio	Descripción
<i>Api para manejar background jobs.</i>	<i>Es utilizado por app-coupons-api 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-api y app-coupons-reports para guardar los logs del stdout de la aplicación.</i>



<b>Interfaz</b>	<i>coupons-api</i>
<b>La implementa</b>	<i>app-coupons-api</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Evaluación de promoción.</i>	<p><i>Pre-requisito: Se debe contar con una clave de aplicación válida.</i></p> <p><i>Es utilizado para evaluar si corresponde aplicar una promoción o no.</i></p>

<b>Interfaz</b>	<i>coupons-service</i>
<b>La implementa</b>	<i>app-coupons-web</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Api que expone Coupons.</i>	<i>Expone los diferentes servicios de Coupons a los usuarios e implementa una interfaz de usuario para facilitar su uso.</i>

<b>Interfaz</b>	<i>gateway-service</i>
<b>La implementa</b>	<i>app-webproxy</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Api Gateway de Coupons.</i>	<i>Api que rutea los request que se realizan desde el Front-End.</i>

<b>Interfaz</b>	<i>kong-admin</i>
<b>La implementa</b>	<i>app-webproxy</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Api de configuración de administrador.</i>	<i>Api que expone Kong para hacer configuraciones del Api Gateway.</i>

<b>Interfaz</b>	<i>queue-service</i>
<b>La implementa</b>	<i>app-amqp</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Servicio de cola de mensajería.</i>	<i>Provee un servicio de cola de mensajería.</i>

<b>Interfaz</b>	<i>amqp-service</i>
<b>La implementa</b>	<i>amqp-server</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Servicio de RabbitMQ.</i>	<i>Servicio que provee acceso al servidor RabbitMQ.</i>

<b>Interfaz</b>	<i>reports-api</i>
<b>La implementa</b>	<i>app-coupons-reports</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Reporte de promoción de uso.</i>	<i>Pre-requisito: Se debe contar con una clave de aplicación válida.</i>  <i>Es utilizado para ver el reporte de uso de una promoción.</i>
<i>Reporte demográfico de promoción.</i>	<i>Pre-requisito: Se debe contar con una clave de aplicación válida.</i>  <i>Es utilizado para ver el reporte demográfico de una promoción.</i>

<b>Interfaz</b>	<i>mongo-service</i>
<b>La implementa</b>	<i>mongodb-server</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Api para manejar MongoDB.</i>	<i>Es utilizado por app-coupons-reports para guardar y recolectar los datos que necesitan ser persistidos de la app-coupons-reports.</i>

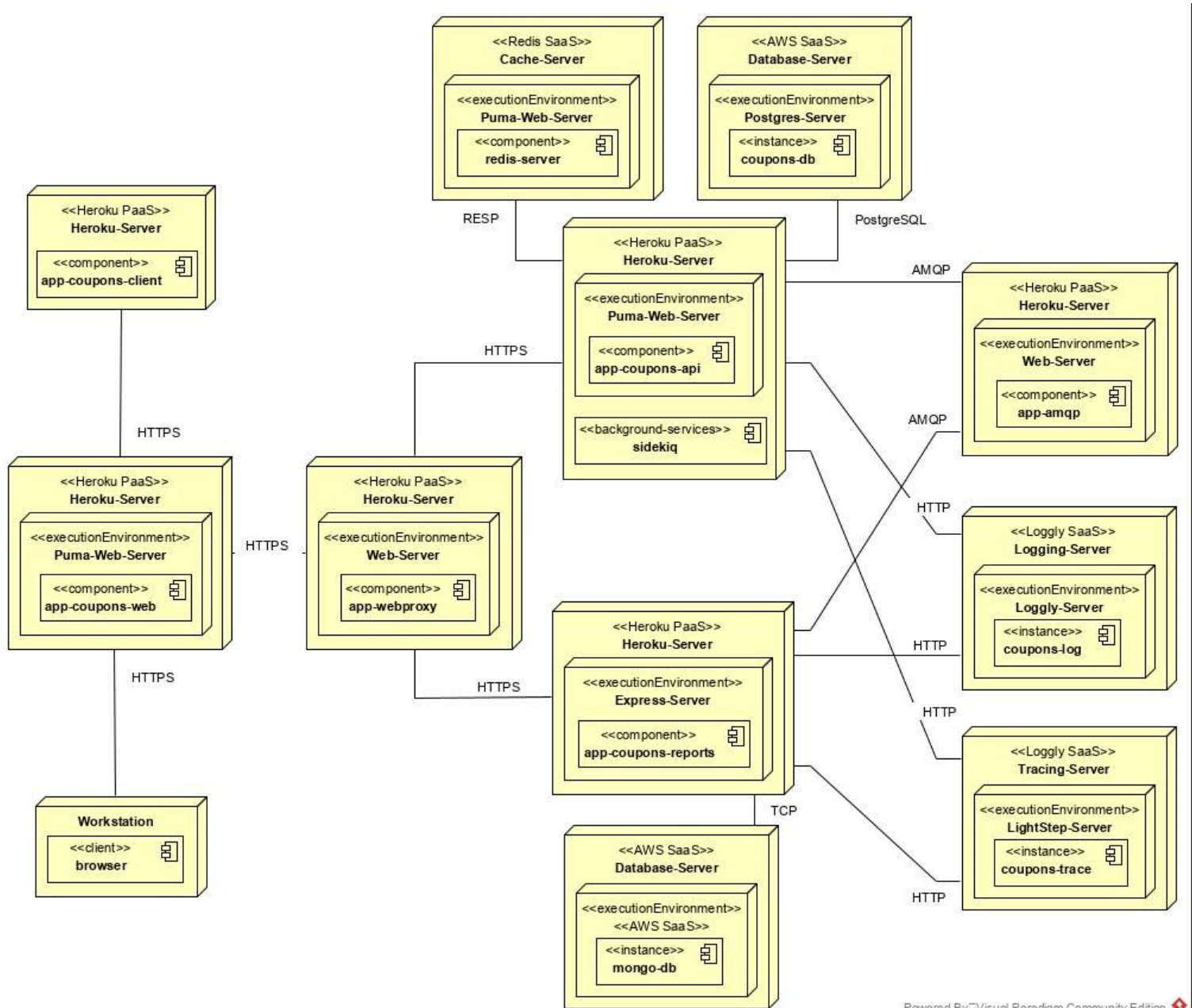
<b>Interfaz</b>	<i>lightstep-api</i>
<b>La implementa</b>	<i>lightstep-server</i>
<b>Servicio</b>	<b>Descripción</b>
<i>Api para guardar la trazabilidad.</i>	<i>Es utilizado por app-coupons-api y app-coupons-reports para guardar la trazabilidad de los Endpoints.</i>

### 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 las aplicaciones y se conecta con diferentes servicios.</i>
<i>Puma-Web-Serv er</i>	<i>N/A</i>	<i>Servidor Web que recibe las peticiones en los Endpoints de las aplicaciones en Ruby.</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 conexiones</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>
<i>Tracking-Server</i>	<i>N/A</i>	<i>Servidor de Tracking, perteneciente a LightStep, que contiene el servidor de LightStep utilizado por nuestro sistema.</i>
<i>LightStep-Server</i>	<i>1.000.000 mensajes/mes, 20 conexiones concurrentes</i>	<i>Servidor de LightStep usado por app-coupons-reports y app-coupons-api para realizar la trazabilidad de las operaciones.</i>
<i>Express-Server</i>	<i>N/A</i>	<i>Servidor Web que recibe las peticiones en los Endpoints de las aplicaciones NodeJs.</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>Clientes Redis se comunican con Redis Server mediante este protocolo.</i>
<i>PostgreSQL</i>	<i>5432/TCP</i>	<i>Clientes PostgreSQL se comunican con el servidor PostgreSQL mediante este protocolo.</i>
<i>AMQP</i>	<i>5671/TCP</i>	<i>Se utiliza para conectar los clientes de colas de mensajería con el servidor de RabbitMQ.</i>

### 3.3.3. Relación con componentes

Nodo	Componente
<i>Heroku-Server</i>	<i>app-coupons-api sidekiq</i>
<i>Heroku-Server</i>	<i>app-coupons-client</i>
<i>Heroku-Server</i>	<i>app-webproxy</i>
<i>Heroku-Server</i>	<i>app-coupons-web</i>
<i>Heroku-Server</i>	<i>app-amqp</i>
<i>Heroku-Server</i>	<i>app-coupons-reports</i>
<i>Puma-Web-Server</i>	<i>app-coupons-web</i>
<i>Web-Server</i>	<i>app-webproxy</i>
<i>Web-Server</i>	<i>app-amqp</i>
<i>Cache-Server</i>	<i>redis-server</i>
<i>Puma-Web-Server</i>	<i>redis-server</i>
<i>Database-Server</i>	<i>postgres-server</i>

<i>Database-Server</i>	<i>mongo-db</i>
<i>Postgres-Server</i>	<i>coupons-db</i>
<i>Logging_Server</i>	<i>coupons-log</i>
<i>Loggly_Server</i>	<i>coupons-log</i>
<i>Tracing-Server</i>	<i>coupons-trace</i>
<i>LightStep-Server</i>	<i>coupons-trace</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 <a href="#">RNF6: Código Fuente</a>.</i>
<i>dotenv-rails</i>	<i>Gestión de variables de ambiente.</i>	<i>Cumplir con el <a href="#">RNF3: Configuración y manejo de secretos</a>. 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 <a href="#">RF6: Baja de promoción</a>. 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 <a href="#">RF4: Gestión de promociones</a> y <a href="#">RF8: Evaluación de promoción</a>.</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 <a href="#">RF5: Listado de Promociones</a>.</i>
<i>image_processing</i>	<i>Permite manejar el procesamiento de imágenes.</i>	<i>Cumplir con la muestra de imagen del <a href="#">RF1: Registro de usuario</a>.</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 <a href="#">RNF2: Confiabilidad y disponibilidad</a>.</i>
<i>loggier</i>	<i>Envía mensajes de log a Loggly.</i>	<i>Cumplir con el <a href="#">RNF8: Identificación de fallas</a>.</i>
<i>sidekiq</i>	<i>Procesa background jobs.</i>	<i>Ayuda a cumplir con el <a href="#">RNF1: Performance</a>.</i>
<i>jwt</i>	<i>Provee mecanismos para gestionar 'JWT's.</i>	<i>Cumplir con el <a href="#">RNF4: Autenticación y Autorización</a></i>
<i>fast_jsonapi</i>	<i>A lightning fast JSON:API serializer for Ruby Objects.</i>	<i>Ayuda a cumplir con el <a href="#">RNF9: Independencia de aplicaciones</a></i>
<i>faraday</i>	<i>Faraday is an HTTP client library that provides a common interface over many adapters</i>	<i>Ayuda a cumplir con el <a href="#">RNF9: Independencia de aplicaciones</a></i>

<i>serializer</i>	<i>Serializer is a Ruby on Rails tool for adding accessor to serialized attributes with support for types and defaults.</i>	Ayuda a cumplir con el <a href="#"><u>RNF9: Independencia de aplicaciones</u></a>
<i>bunny</i>	<i>Bunny is a RabbitMQ client that focuses on ease of use. It is feature complete, supports all recent RabbitMQ features and does not have any heavyweight dependencies.</i>	Ayuda a cumplir con el <a href="#"><u>RNF9: Independencia de aplicaciones</u></a>

## 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:

'Logglie': 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 'logglie'`

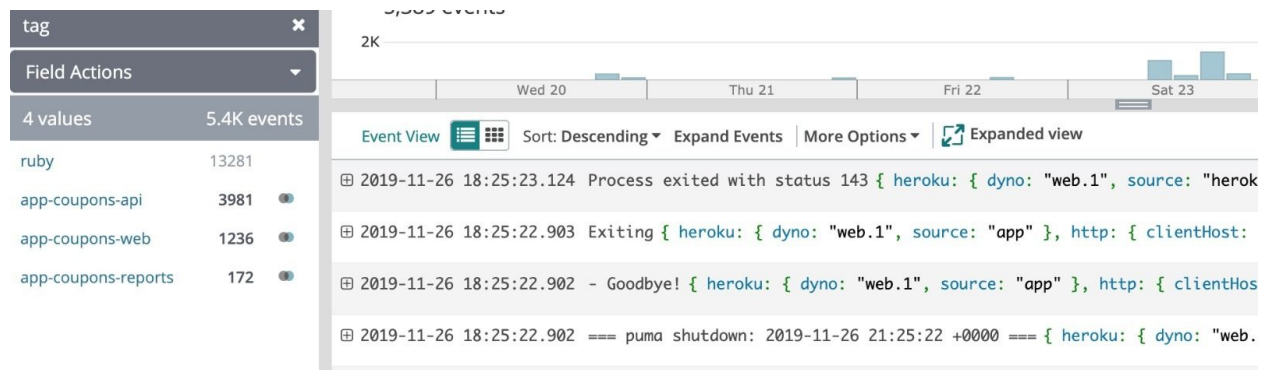


- `config.logger = Logglier.new("https://logs-01.loggly.com/inputs/TOKEN/tag/ruby/", :threaded => true)`

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`

Imagen de Loggly en producción:



#### 4.4.9. RNF9: Independencia de aplicaciones

Para poder hacer el cambio a Microservicios, se realizaron los siguientes pasos:

##### **1. Desarrollo de toda la aplicación en un monolito.**

El desarrollo fue producto del primer obligatorio entregado, en el cual se había realizado los requerimientos base del software.

Deploy del Monolito: <https://app-coupons.herokuapp.com/>

##### **2. Separación del Front-End y el Back-End del monolito, a dos aplicaciones en Ruby.**

Al ser requerida la independencia de aplicaciones, como primer paso decidimos realizar la separación del monolito existente, en 2 servicios por separado.

Por un lado, el Front-End que había sido realizado en Ruby y por otro lado el Back-End que había sido realizado en Ruby también.

El principal inconveniente que surgió, fue que las vistas estaban extremadamente acopladas a la lógica de los servicios, por esta razón, se tuvo que hacer uso de diferentes técnicas y gemas para llevar a cabo este proceso de separación:

- Se utilizó la gema Faraday, la cual sirvió para hacer peticiones entre el Front-End y el Back-End a través de JSON / HTTP, ya que ahora no se podían usar llamadas directas.
- Se utilizó la gema Serializer, la cual sirvió para acceder a atributos serializados que ayudaron a mejorar el tiempo de mensajería entre el Front-End y el Back-End.

Deploy del Back-End: <https://app-coupons-api.herokuapp.com/>

Deploy del Front-End: <https://app-coupons-web.herokuapp.com/>

##### **3. Separación del servicio de Reportería del monolito a una aplicación NodeJs.**

Se separó el servicio de repostería creando una aplicación NodeJs, la cual cumplía con los RF7: Reporte de uso y RF15: Reporte demográfico de usuarios.

Una vez finalizado el servicio, se procedió a eliminar el servicio de reportería que se encontraba alojado en el servicio en Ruby.

En el Front-End, se pasó a apuntar al nuevo servicio de reportería, dejando sin efecto el anterior en Ruby.

El Deploy está en la siguiente URI: <https://app-coupons-reports.herokuapp.com/>

#### **4. Configuración y Deploy de Kong como Api Gateway a Heroku.**

Se realizó el Deploy de Kong a Heroku, para poder utilizar el mismo como punto de entrada a los servicios del sistema.

Por el momento, sólo será utilizado por el Front-End en Ruby, pero por letra, se especifica que próximamente existirá un BackOffice el cual se conectará a los servicios a través de Kong.

La configuración se realizó a través de la API que expone KONG en el End-Point /kong-admin. Se adjuntará en la entrega, un archivo exportado de Postman con la configuración realizada.

Para todos los Endpoints, se configuró que tanto el 'connect\_timeout', como 'read\_timeout' y 'write\_timeout' fueran de 60 segundos.

También se estableció que los reintentos luego de haber existido una falla en redirigir una llamada a un servicio, fuera de 5 intentos.

El Deploy está en la siguiente URI: <https://app-webproxy.herokuapp.com/>

#### **5. Configuración y Deploy de RabbitMQ como servicio a Heroku.**

Se utilizó RabbitMQ para gestionar mensajería asincrónica utilizando colas de mensajes.

Para realizar el Deploy en Heroku, se utilizó CloudAMQP, para facilitar la integración de RabbitMQ a Heroku.

El Deploy está en la siguiente URI: <https://app-amqp.herokuapp.com/>

## **6. Integración de las aplicaciones con Kong como Api Gateway.**

Las llamadas entre el servicio de Front-End con los servicios del Back-End, son realizadas a través del API Gateway, nunca directamente una con la otra.

De esta forma, se logra tener un punto de independencia en el cual configurando el Api Gateway de diferentes maneras, se pueden hacer diversas estrategias sin tener que modificar el Front-End que suele ser lo más difícil de modificar.

## **7. Integración del Back-End en Ruby con la aplicación de Repostería en NodeJs a través de RabbitMQ.**

Se dejó de enviar los mensajes a través de una REST Api y se pasó a utilizar RabbitMQ para la mensajería entre ambos servicios.

Luego que el servicio de evaluación de promociones, realiza una evaluación, publica el resultado de la misma a la cola de mensajería. Por el otro lado, desde el servicio de repostería, se consume de la cola de mensajería los resultados de evaluación que hayan pendientes.

## **8. Creación de un cliente con el SDK integrado para evaluación de promociones.**

Se creó un cliente en NodeJs, integrado con el código del SDK, sin estar aún separado en un SDK.

De esta forma, se logró confirmar, que el flujo fuera correcto y que el cliente pueda hacer evaluaciones de promociones de manera correcta utilizando el código del SDK internamente.

El Deploy está en la siguiente URI: <https://app-coupons-client.herokuapp.com/>

## **9. Separación del SDK de evaluación de promociones del cliente.**

Una vez el cliente con el SDK fue probado y las pruebas fueron satisfactorias, se procedió a separar el SDK del cliente SDK, para poder tener la independencia del SDK que se pedía en el [RF10: SDK de evaluación de promociones](#).

El SDK está en la siguiente URI: <https://www.npmjs.com/package/coupons-sdk>

## **10. Configuración de ‘LightStep’ SaaS de Tracing.**

Se decidió utilizar como software de Tracing entre los microservicios a LightStep. El mismo es un proveedor de Tracing SaaS, el cual admite comunicarse a través de OpenTracing (SDK de tracing para acoplar a los diferentes servicios).

Se creó un usuario en modalidad gratis, ya que cumplía con los requerimientos solicitados en el [RNF10: Monitoriabilidad](#).

El servicio de Light Step provee tanto trazabilidad entre diferentes servicios, como el tiempo de latencia de las respuesta y la cantidad de operaciones por minuto que reciben cada uno de los Endpoints del sistema.

## **11. Integración entre el Back-End en Ruby y la aplicación en NodeJs con LightStep a través del protocolo OpenTracing.**

Primero se integró el componente de reportería que fue desarrollado en NodeJs, el cual reporta bajo el servicio app-coupons-reporting con sus respectivos Endpoints en el sistema de LightStep.

Se utilizaron los siguientes paquetes externos:

- Opentracing: SDK que brinda las funcionalidades principales para comunicarse con LightStep.
- Lightstep: Paquete utilizado para poder integrar LightStep para ser el tracer utilizado por Opentracing.

En una segunda instancia se integró el componente de Back-End en Ruby con Opentracing para conectarse a LightStep.

## 4.4.10. RNF10: Monitoreabilidad

### **app-coupons-reports**

Dependencias externas utilizadas:

- 'opentracing': Se utilizó como protocolo por defecto de comunicación de tracing.
- 'lightstep-tracer': Se utilizó como tracer para que opentracing pueda conectarse con el SaaS de LightStep.

### **app-coupons-api**

Dependencias externas utilizadas:

- 'lightstep-tracer': Se utilizó como tracer para que opentracing pueda conectarse con el SaaS de LightStep.

### **Justificación de la elección**

Se agregó tracing a nivel de los endpoints de reportería para cumplir con los siguientes requerimientos:

- Poder notificar la cantidad de operaciones por minuto que recibe el endpoint en cuestión.
- Poder determinar el tiempo de respuesta en promedio que demora el endpoint en resolver el requerimiento.
- Poder determinar la trazabilidad de una operación a través de los microservicios, utilizando un transaction ID determinado por opentracing.

## 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 ActiveSupport 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 está `'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.

#### 4.5.9. RF9: Gestión de usuarios

**Servicio:**

app-coupons-api

**Lenguaje:**

Ruby

**Implementación:**

- A 'organization' se borró 'belongs\_to\_admin'
- A 'users' se le agregó un campo 'role', que puede tener los siguientes valores:
  - 'normal': Son los usuarios de la organización.
  - 'admin': Son los usuarios administradores de la organización.
  - 'finance': Son los usuarios financieros de la organización.
- Se agregó como 'tentant' de 'promotion\_definition', la 'organization' a la cual pertenece la promoción, para evitar que un 'admin' pueda ingresar a una 'promotion\_definition' de otra 'organization'.
- En la creación de usuario, se agregó un 'combobox' para poder seleccionar los roles a invitar por mail.

## 4.5.10. RF10: SDK de evaluación de promociones

### Servicio:

app-coupons-reports

### Lenguaje:

NodeJs

### Packages utilizados:

*'request'*: Está diseñado para ser la forma más simple posible para hacer llamadas HTTP. Soporta HTTPS y sigue redirecciones por defecto.

### Implementación:

Se creó un método llamado *'requestevaluationjsonformat'* que recibe un json con todos los parámetros necesarios de la interfaz para hacer el request.

Se creó un método llamado *'requestevaluationseveralparamsformat'* que recibe 9 parámetros siendo 3 opcionales:

- *'token'*: Es la clave de aplicación.
- *'promo\_code'*: Es el código del *'PromotionDefinition'*, el que viene en la URI.
- *'total'*: Es el valor de la compra.
- *'promo\_type'*: Acepta dos valores, coupon o discount.
- *'txnid\_or\_couponcode'*: Dependiendo si es coupon o discount, va txnid o couponcode.
- *'attr\_list'*: Lista de atributos a evaluar.
- *'country'*: Es opcional, es el país donde se evalúa la promoción.
- *'city'*: Es opcional, es la ciudad en donde se evalúa la promoción.
- *'birthday'*: Es opcional, es el día de nacimiento de la persona que utiliza el cupón.

Se creó un método *'evaluatepromotion'* que no puede ser accedido desde el cliente y sirve como wrapper de la librería que se usa para realizar el request.

Sea que el cliente haya utilizado *'requestevaluationjsonformat'* o *'requestevaluationseveralparamsformat'*, ambas terminan utilizando *'evaluatepromotion'* para contactar con el servicio de evaluación.

Al haber sido realizada la publicación del SDK en npm, se tomaron las siguientes decisiones:

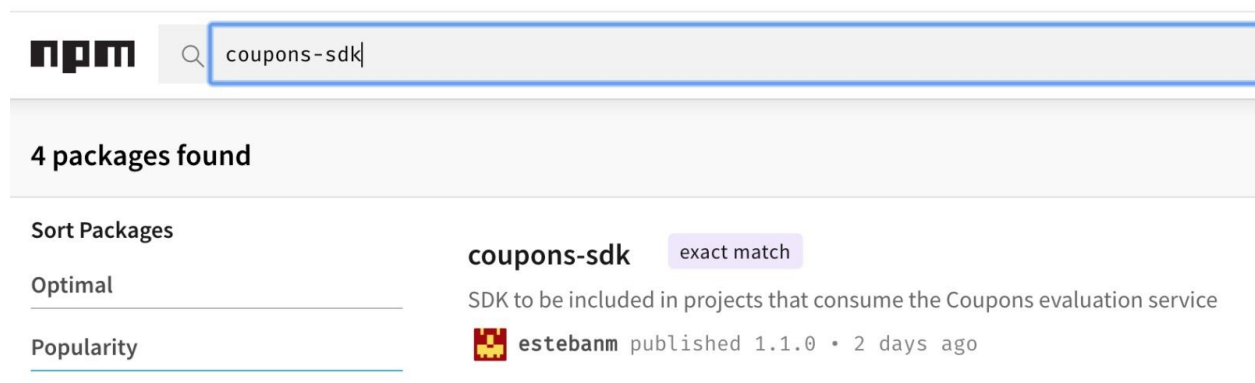
- package.json con la descripción en formato formal.
- package.json con la versión 1.1.0 ya que es la recomendación de npm cuando se hizo un incremento menor en la funcionalidad que todavía sigue siendo compatible con los clientes anteriores. El cambio puntualmente fue el ingreso de los 3 atributos nuevos.

## Incrementing semantic versions in published packages

To help developers who rely on your code, we recommend starting your package version at **1.0.0** and incrementing as follows:

Code status	Stage	Rule	Example version
First release	New product	Start with 1.0.0	1.0.0
Backward compatible bug fixes	Patch release	Increment the third digit	1.0.1
Backward compatible new features	Minor release	Increment the middle digit and reset last digit to zero	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit and reset middle and last digits to zero	2.0.0

Imagen de la publicación en NPM de 'coupons-sdk'



URL en npm del SDK: <https://www.npmjs.com/package/coupons-sdk>



#### 4.5.11. RF11: Límite de uso de cupón

**Servicio:**

app-coupons-api

**Lenguaje:**

Ruby

**Implementación:**

Para que el sistema permitiera múltiples usos de los cupones, se decidió hacer los siguientes cambios:

- Se cambió la variable booleana 'used' de 'Promotion' por un entero 'uses', con la finalidad de permitir múltiples usos de los cupones.
- A nivel de la interfaz de usuario, se agregó un campo para poder especificar cuántos usos tiene el set de cupones que se va a generar.
- Al evaluar una promoción del tipo cupón:
  - Si la misma es correcta y la cantidad de usos del cupón es mayor o igual a 1, se resta 1 a la cantidad de usos disponibles del cupón.
  - Si la cantidad de usos del cupón es igual a 0, se envía un mensaje de error.
- Para las promociones con transacción, se crea una promoción nueva con 0 uses.

## 4.5.12. RF12: Expiración de promociones

### **Servicio:**

app-coupons-api

### **Lenguaje:**

Ruby

### **Gemas utilizadas:**

‘sidekiq-scheduler’: Se agregó esta gema para poder hacer el background job de chequear a las 8 am cada día y enviar los mails a los admin de cada organización.

### **Implementación:**

Se agregó el campo ‘expiration\_date’ a ‘promotion\_definition’.

- Condiciones de ‘expiration\_date’:
  - Valor mínimo: Fecha del día de creación de la promoción.
- Se agregó un worker que corre todos los días a las 8 am.
  - Envía un mail a los admin de las organizaciones diciendo las promociones que se vencieron hoy y las que se van a vencer mañana.
- Cuando se evalúa una promoción de cualquier tipo de promoción:
  - Si ‘expiration\_date’ es menor al día que se realiza la evaluación, se envía un mensaje de error comentando que la promoción está vencida.

### 4.5.13. RF13: Agregar cupones a promoción

**Servicio:**

app-coupons-api

**Lenguaje:**

Ruby

**Gemas utilizadas:**

‘sidekiq’: Se utilizó esta gema para poder generar cupones en background.

**Implementación:**

Cuando se va a agregar cupones a una promoción:

- Se accede a la pantalla de la promoción.
- Si la promoción es del tipo cupón, aparecerá un botón para ir a la sección de generar cupones.
- En ese menú se selecciona:
  - La cantidad de cupones que se desea crear.
  - La cantidad de usos que pueden tener.
  - La fecha de expiración que los cupones van a tener.

#### 4.5.14. RF14: Vencimiento de cupones

**Servicio:**

app-coupons-api

**Lenguaje:**

Ruby

**Implementación:**

Se agregó la columna 'expiration\_date' a 'promotion'.

- Condiciones que debe cumplir el 'expiration\_date':
  - Valor máximo: El valor del 'expiration\_date' de la 'promotion\_definition' a la que pertenece.
  - Valor mínimo: Es la fecha del día que se crea el cupón.
- Cuando se evalúa una promoción del tipo cupón:
  - Si el cupón está vencido, se envía un mensaje de error comentando que el cupón está vencido.

## 4.5.15. RF15: Reporte demográfico de usuarios

### **Servicio:**

app-coupons-reports

### **Lenguaje:**

NodeJs

### **Packages utilizados:**

*'body-parser'*: Parsea body de request entrante desde un middleware antes que llegue a los controladores. Es accesible a través de la propiedad req.body.

*'dotenv'*: Módulo que carga variables de entorno desde un archivo .env a process.env. Guardando la configuración en un entorno separado del código, basado en la metodología The Twelve-Factor App.

*'express'*: Web framework rápido, sin opiniones y minimalista.

*'jsonwebtoken'*: Implementación de JSON Web Tokens.

*'mongoose'*: Es una herramienta de modelado de MongoDB designada a trabajar en ambientes asincrónicos.

*'request'*: Está diseñado para ser la forma más simple posible para hacer llamadas HTTP. Soporta HTTPS y sigue redirecciones por defecto.

*'amqpLib'*: Cliente AMQP para NodeJs, el cual es un protocolo de mensajería abierto y de propósito general, usado para comunicarse con RabbitMQ.

*'mocha'*: Mocha es un framework de testing rico en features para JavaScript que corre sobre NodeJs, haciendo que los testing asíncronos sean fáciles.

*'moment'*: Se utiliza para validar el formato de fecha para que respete lo especificado en la letra.

## **Implementación:**

La implementación de este requerimiento, no tuvo impacto en cómo es autenticada la clave de aplicación. Ya que cuando se consume el reporte nuevo, la validación se delega al mismo sector que evalúa para el reporte anterior.

Se consume de la cola de mensajería RabbitMQ, que es publicada por el servicio de cupones una vez se evalúa una promoción.

Los datos de la evaluación son guardados en la base de datos.

Si entre ellos están los dato de:

- 'country': Este será almacenado en la base de datos.
- 'city': Este será almacenado en la base de datos.
- 'birth\_date': Si este campo está presente y respeta el formato, entonces se precalcula el dato de la edad para agilizar la generación del reporte futuro.

## 5. Proceso de desarrollo, testing y deployment

### 5.1. Instrucciones para el desarrollo

#### 5.1.1. app-coupons-reports

- Download repository

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

- Package dependencies

```
npm install
```

- Crear archivo .env en la raíz del proyecto con los siguientes parámetros

```
DB_URL = ...  
SECRET_KEY_BASE = ...
```

#### 5.1.2. app-coupons-web

- Download repository

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

- Gemfile dependencies

```
bundle install
```

### 5.1.3. app-coupons-api

- Download repository

```
git clone https://github.com/ArqSoftPractica/ASP_Obl_Back.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 Logglier to config/environments/.rb

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

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

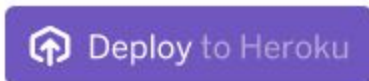


#### 5.1.4. app-webproxy

- Ir al repositorio de heroku-kong

<https://github.com/heroku/heroku-kong>

- Clickear el botón Deploy to Heroku



- Asignar como nombre de la aplicación app-webproxy
- Clickear Deploy app

#### 5.1.5. app-amqp

- Login en Heroku via terminal

```
heroku login
```

- En una terminal de comandos

```
heroku create -a app-amqp
```

- Asignar el Addon CloudAMQP a la aplicación creada

```
heroku addons:create cloudamqp:lemur
```

### 5.1.6. coupons-sdk

- Download repository

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

- Package dependencies

```
npm install
```

### 5.1.7. app-coupons-client

- Download repository

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

- Package dependencies

```
npm install
```

## 5.2. Instrucciones para las pruebas

### 5.2.1. En las aplicaciones Ruby:

- 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.

### En NodeJs:

- Pruebas automáticas:
  - Requerimientos: Tener instalado el paquete 'mocha'.
  - Especificación: Los test se escriben en una carpeta test en la raíz.

```
npm test
```

## 5.3. Instrucciones para el deploy

### 5.3.1. app-coupons-reports

- Descargar el repositorio

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

- Instalar las dependencias de package.json

```
npm install
```

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

```
heroku login
```

- Crear un nuevo repositorio Heroku para la aplicación

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

- Configurando el 'buildpack' por defecto de la aplicación

```
heroku buildpacks:set heroku/node
```

- Haciendo 'Push' al repositorio Heroku.

```
git push heroku master
```

- Agregando las variables de ambiente en Heroku

```
DB_URL = ...  
SECRET_KEY_BASE = ...
```

### 5.3.2. app-coupons-api

- Descargar el repositorio

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

- Instalar las dependencias de Gemfile

```
bundle install
```

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

```
heroku login
```

- Crear un nuevo repositorio Heroku para la aplicación

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

- 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.

### 5.3.3. app-coupons-web

- Descargar el repositorio

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

- Instalar las dependencias de Gemfile

```
bundle install
```

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

```
heroku login
```

- Crear un nuevo repositorio Heroku para la aplicación

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

- 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

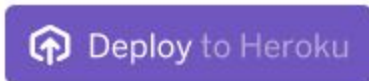
```
API_URL = ...  
LANG = en_US.UTF-8  
RACK_ENV = production  
RAILS_ENV = production  
RAILS_LOG_TO_STDOUT = enabled  
RAILS_SERVE_STATIC_FILES = enabled  
SECRET_KEY_BASE = ...
```

### 5.3.4. app-webproxy

- Ir al repositorio de heroku-kong

<https://github.com/heroku/heroku-kong>

- Clickear el botón Deploy to Heroku



- Asignar como nombre de la aplicación app-webproxy
- Clickear Deploy app

### 5.3.5. app-amqp

- Login en Heroku via terminal

```
heroku login
```

- En una terminal de comandos

```
heroku create -a app-amqp
```

- Asignar el Addon CloudAMQP a la aplicación creada

```
heroku addons:create cloudamqp:lemur
```



### 5.3.6. coupons-sdk

- Download repository

```
git clone https://github.com/ArgSoftPractica/ASP\_Obl\_SDK.git
```

- Package dependencies

```
npm install
```

- Creación de cuenta en npm
- Login en npm via terminal

```
npm login
```

- Pararse en la carpeta root del proyecto y realizar

```
npm publish
```

```
→ ASP_Obl_SDK git:(develop) npm publish
npm notice
npm notice 📦  coupons-sdk@1.1.0
npm notice === Tarball Contents ===
npm notice 331B  package.json
npm notice 2.0kB  index.js
npm notice 13B   README.md
npm notice 1.1kB services/evaluation.service.js
npm notice === Tarball Details ===
npm notice name:           coupons-sdk
npm notice version:        1.1.0
npm notice package size:   1.3 kB
npm notice unpacked size:  3.5 kB
npm notice shasum:         744d222e06b1bb1b0926569c0743b09770a52099
npm notice integrity:      sha512-Wnh0mqDtopAXc[...]5Xpy/p7IBmACg==
npm notice total files:    4
npm notice
+ coupons-sdk@1.1.0
```

### 5.3.7. app-coupons-client

- Download repository

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

- Instalar las dependencias de package.json

```
npm install
```

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

```
heroku login
```

- Crear un nuevo repositorio Heroku para la aplicación

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

- Configurando el 'buildpack' por defecto de la aplicación

```
heroku buildpacks:set heroku/node
```

- Haciendo 'Push' al repositorio Heroku.

```
git push heroku master
```

## 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

#### 7.1.1. app-coupons-api

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

## 7.1.2. app-coupons-reports

### Authorization:

Test	Resultado
should throw an InvalidCredentials exception when the token is not present	Ok
should throw an InvalidCredentials exception when the token is not valid	Ok
should throw a InvalidDataReceived exception when the promoId does not exists	Ok
should throw a forbidden exception when the promoId is not allowed for that user	Ok
Should authorize when everything goes well	Ok

### Statistic Summary:

Test	Resultado
should return an empty summary when the promoId does not exists.	Ok
should return loaded summary and data validates ok	Ok

### Demographic Report:

Test	Resultado
should return {count: 0} when the promoId does not exists	Ok
should return {count: total promo requests} when the promoId exists and receive unspecified filter	Ok
should return {count: total promo requests for that country} when the promoId exists and receive country filter	Ok
should return {count: total promo requests for that city} when the promoId exists and receive city filter	Ok
should return {count: total promo requests for that age range} when the promoId exists and receive an age range filter	Ok
should return {count: total promo requests for that complete filter} when the	Ok

promoId exists and receive country, city and an age range filter	
--	--

## Imagen de la ejecución de las pruebas

```
→ ASP_Obl_Reports git:(develop) ✖ npm test

> coupons-reports@1.0.0 test /Users/estebanmuzio/Desktop/Coupons-Stats/ASP_Obl_Reports
> mocha

Authorization
  isAuthorized()
    ✓ should throw an InvalidCredentials exception when the token is not present
    ✓ should throw an InvalidCredentials exception when the token is not valid
    ✓ should throw a InvalidDataReceived exception when the promoId does not exists (165ms)
    ✓ should throw a Forbidden exception when the promoId is not allowed for that user (169ms)
    ✓ should authorize when everything goes well (156ms)

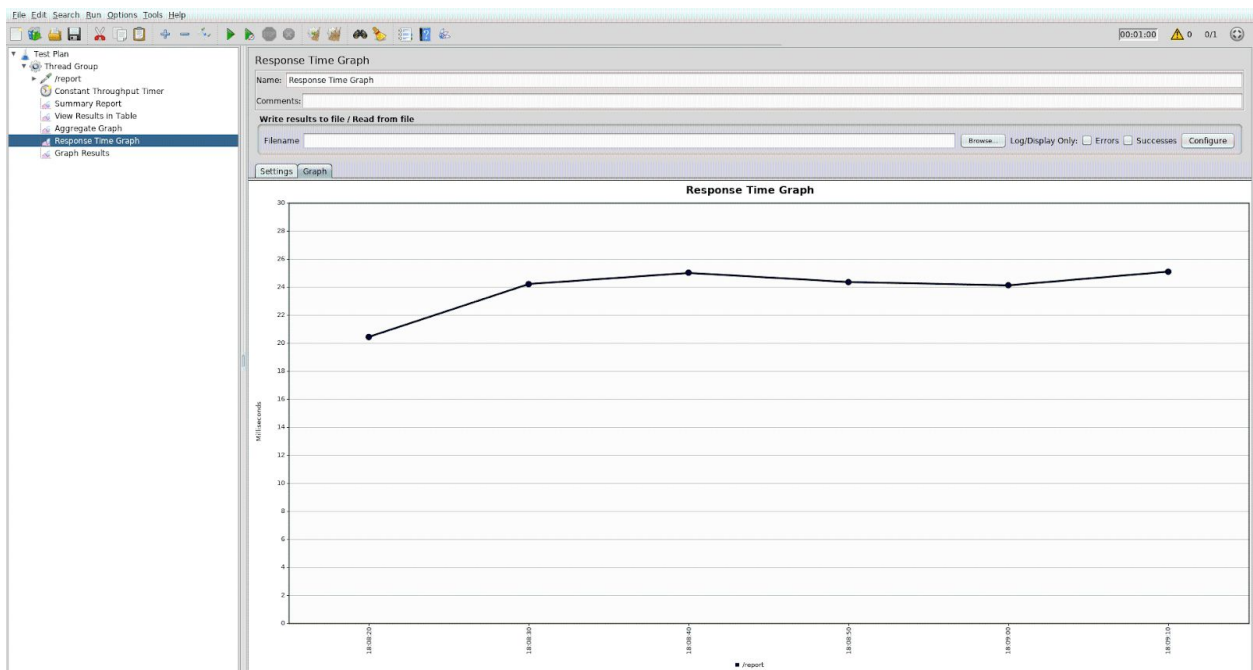
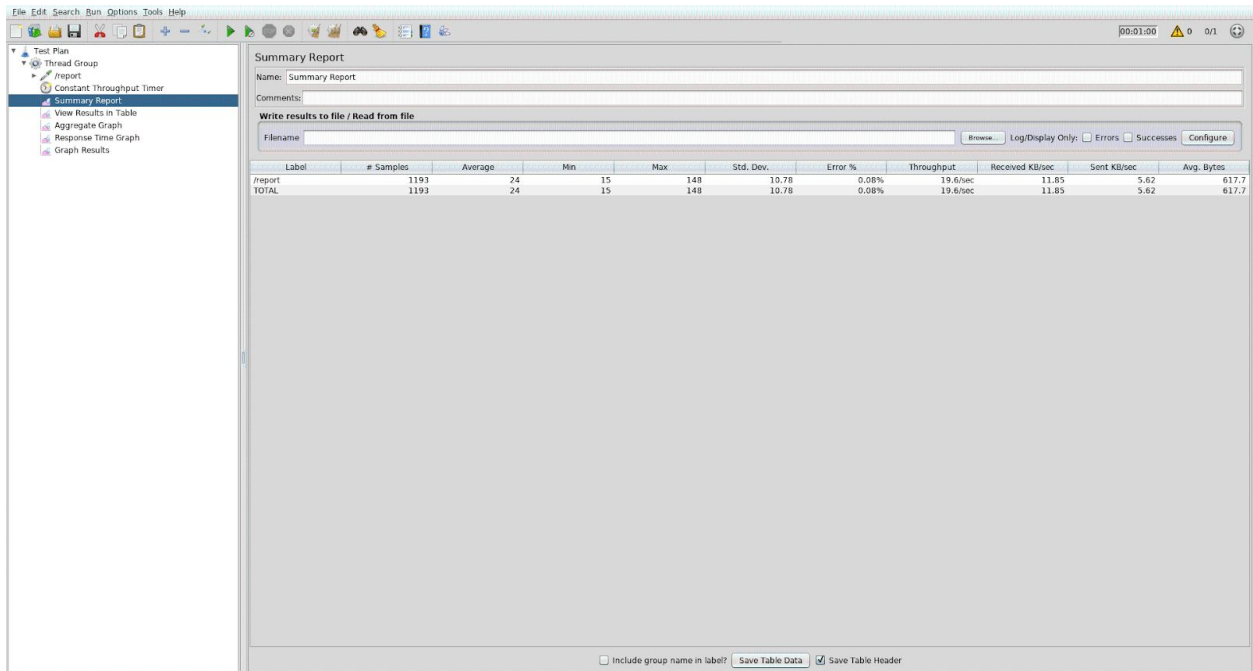
Statistics Summary
  getPromoStatsSummary()
    ✓ should return an empty summary when the promoId does not exists (159ms)
    ✓ should return loaded summary and data validates ok (160ms)

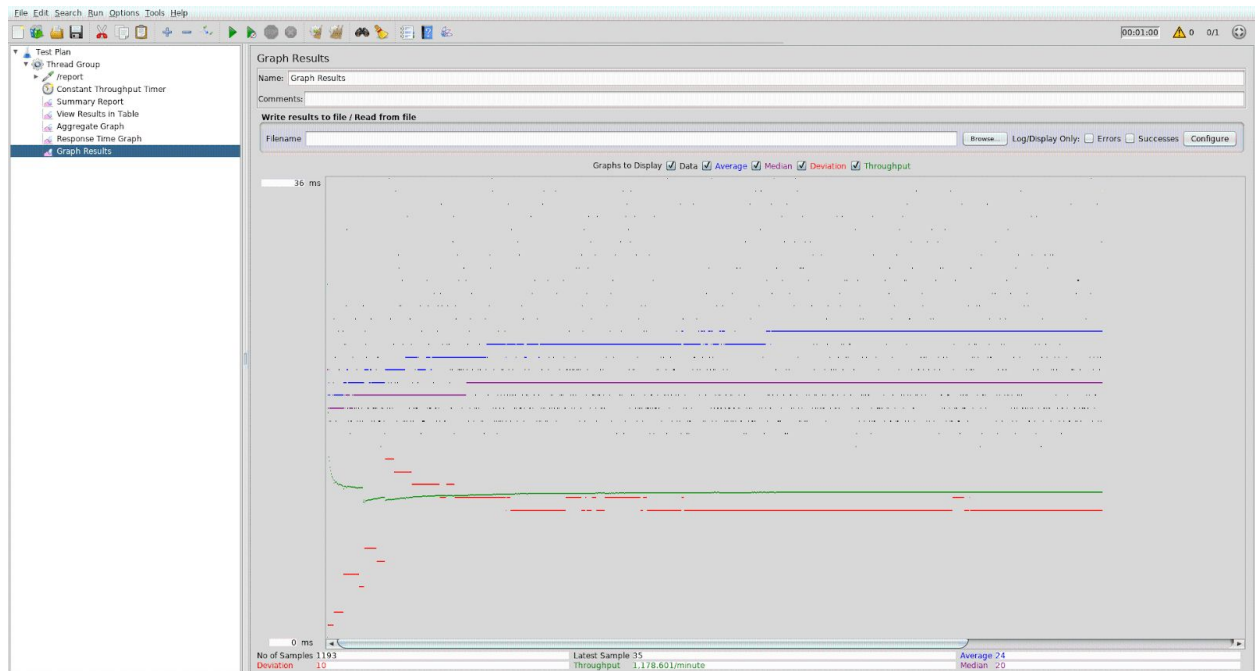
Demographic Report
  getPromoStatsDemographic()
    ✓ should return {count: 0} when the promoId does not exists (158ms)
    ✓ should return {count: total promo requests} when the promoId exists and receive unspecified filter (160ms)
    ✓ should return {count: total promo requests for that country} when the promoId exists and receive country filter (161ms)
    ✓ should return {count: total promo requests for that city} when the promoId exists and receive city filter (158ms)
    ✓ should return {count: total promo requests for that age range} when the promoId exists and receive an age range filter (158ms)
    ✓ should return {count: total promo requests for that complete filter} when the promoId exists and receive country, city and an age range filter (160ms)

13 passing (7s)
```

## 7.2. Pruebas de carga

Con 'throughput' constante:

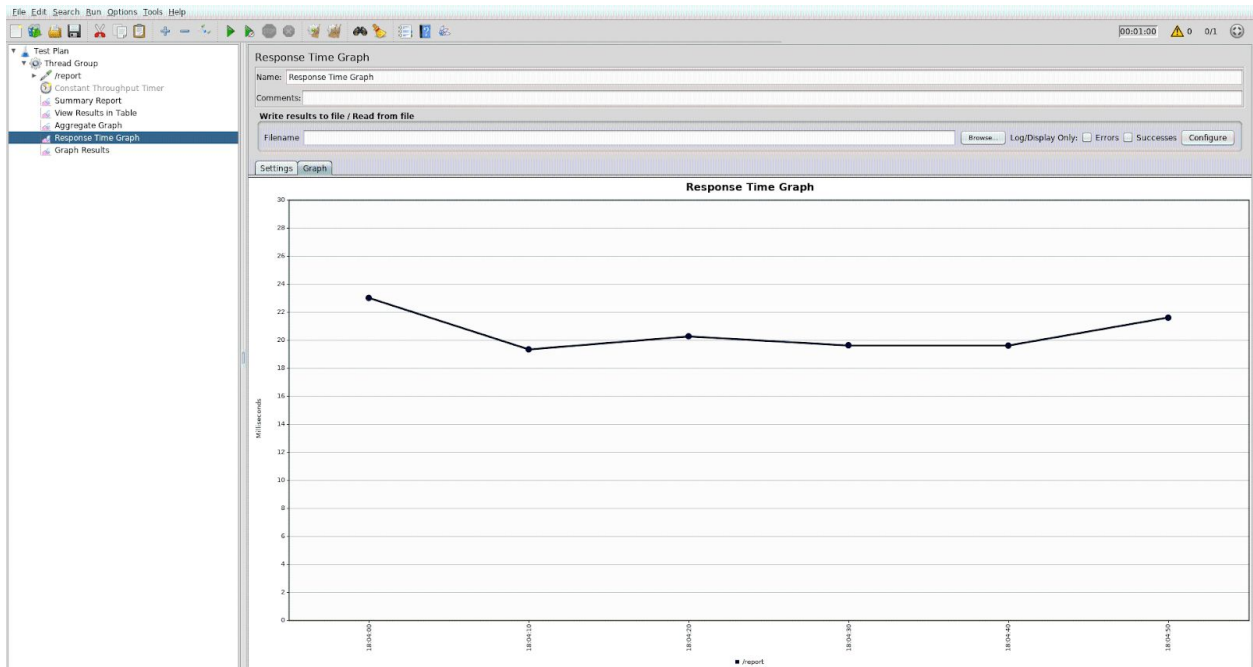
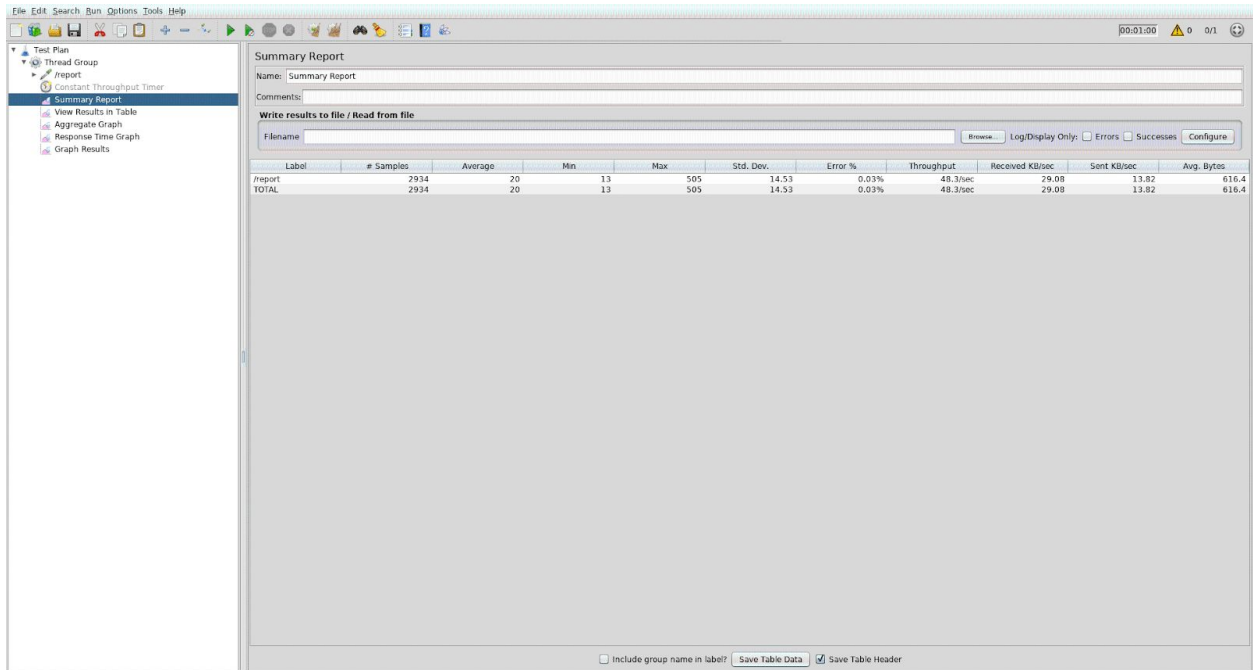


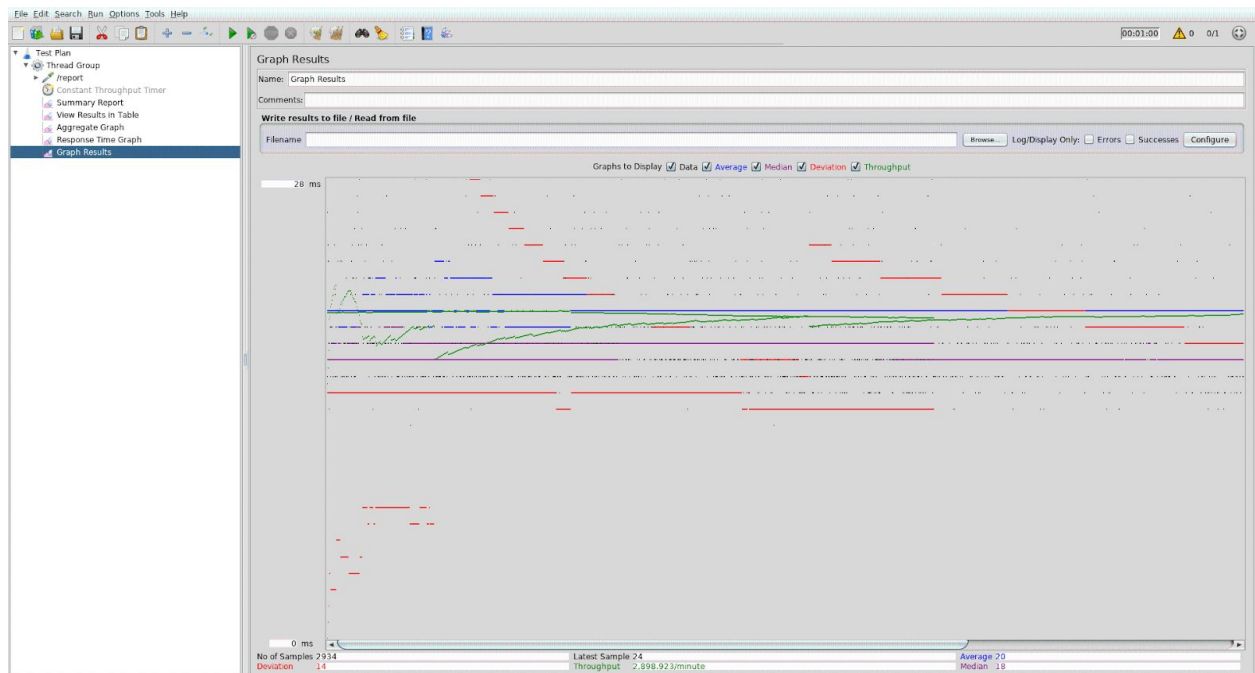


## Resultados:

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

Con 'throughput' no constante:





## 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:

ArqSoftPractica / ASP\_Obl Private

[Code](#) [Issues 0](#) [Pull requests 0](#) **[Projects 1](#)**

**1 Open** ✓ **0 Closed**

**Coupons** Private No description

Updated 14 hours ago

## 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 matiashrndz opened this issue 22 days ago · 1 comment



matiashrndz 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

 Closed matiashrndz opened this issue 25 days ago · 1 comment



matiashrndz 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 1 - 10 de Octubre del 2019	
Card	Criterios de aceptación
<a href="#"><i>RNF1. Performance</i></a>	1 de 1
<a href="#"><i>RNF2. Reliability and Availability</i></a>	4 de 4
<a href="#"><i>RNF3. Configuration and secret's handling</i></a>	1 de 1
<a href="#"><i>RNF4. Authentication and Authorization</i></a>	4 de 4
<a href="#"><i>RNF5. Security</i></a>	1 de 1
<a href="#"><i>RNF6. Source Code</i></a>	7 de 7
<a href="#"><i>RNF7. Tests</i></a>	2 de 2
<a href="#"><i>RNF8. Identify faults</i></a>	2 de 2
<a href="#"><i>RF1. Signup via Invitation Link</i></a>	6 de 6
<a href="#"><i>RF1. Signup via Web</i></a>	6 de 6
<a href="#"><i>RF2. Signin</i></a>	4 de 4
<a href="#"><i>RF3. Managing Application Key</i></a>	2 de 2
<a href="#"><i>RF4. Register Promotion</i></a>	1 de 1
<a href="#"><i>RF5. Show Promotions List</i></a>	6 de 6
<a href="#"><i>RF6. Deregister Promotion</i></a>	1 de 1
<a href="#"><i>RF7. Promotions Reports</i></a>	7 de 7
<a href="#"><i>RF8. Promotion Evaluation</i></a>	4 de 4

Release 2 - 28 de Noviembre del 2019	
Card	Criterios de aceptación
<a href="#"><i>RNF9. Application Independence</i></a>	<i>1 de 1</i>
<a href="#"><i>RNF10. Monitoring</i></a>	<i>1 de 1</i>
<a href="#"><i>RF9. User Management</i></a>	<i>6 de 6</i>
<a href="#"><i>RF10. Promotion Evaluation SDK</i></a>	<i>2 de 2</i>
<a href="#"><i>RF10. Promotion Evaluation Client SDK</i></a>	<i>1 de 1</i>
<a href="#"><i>RF11. Coupon use limit</i></a>	<i>2 de 2</i>
<a href="#"><i>RF12. Promotions expiration</i></a>	<i>3 de 3</i>
<a href="#"><i>RF13. Add coupons to promotion</i></a>	<i>2 de 2</i>
<a href="#"><i>RF14. Coupons expiration</i></a>	<i>2 de 2</i>
<a href="#"><i>RF15. Demographic User Report</i></a>	<i>1 de 1</i>

## 9. Detalle de Endpoints

La aplicación cuenta con tres 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
<p>Header: Authorization: token</p> <p>Body:</p> <pre>{   "code": "promo_code",   "metadata": {     "total": int,     "transaction_id": "id",     "country": string ,     "city": string     "birthday": string     "attributes": { "attr1":int, "attr2":int }} }</pre> <p><b>Nota:</b> transaction_id podría sustituirse por coupon_code si se trata de una promo tipo Coupon. Los miembros country (ISO3166-1), city (IATA) y birthday son todos opcionales.</p>	<p>(200) - si pudo evaluar ok retorna { "success": "Valid promotion" }</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>	<p>(404) - Not Found</p>

## Reporte de Uso

<b>/api/promo_stats/{:id}/summary</b>	<b>GET</b>	<b>POST, PUT, DELETE</b>
<p>Header: Authorization: token</p> <p>Body: null</p> <p>id: es el identificador de la promoción sobre la cual se solicita el reporte</p>	<p><b>(200)</b> - retorna el reporte de uso de la promoción cuyo código recibe como parte de la uri</p> <p><b>(400)</b> - Bad Request, no se encuentra promoción en el sistema cuyo id concuerde con el especificado en la uri</p> <p><b>(401)</b> - Unauthorized, el request llega sin token o el token es incorrecto</p> <p><b>(403)</b> - Forbidden, el código de la promoción con id igual al especificado en la uri, no figura dentro de los códigos que el usuario puede consultar y que figuran en el JWT</p>	<p><b>(404)</b> - Not Found</p>

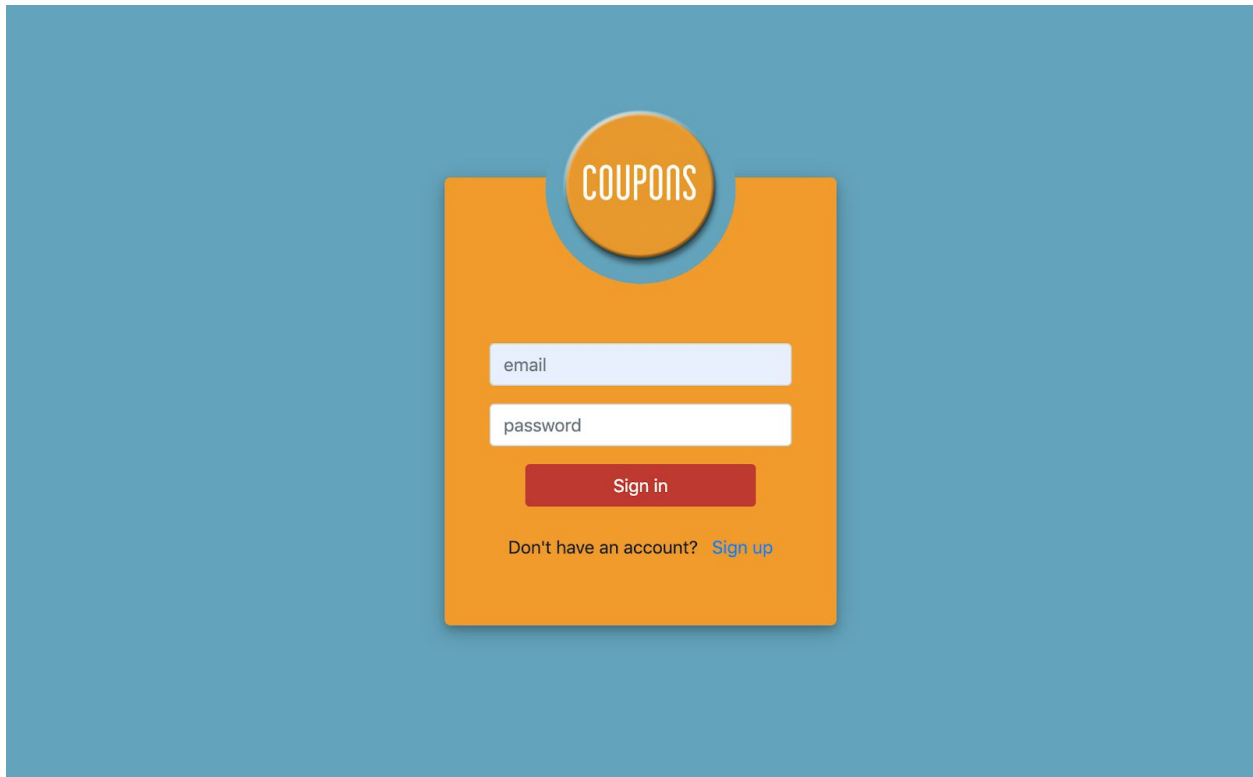
## Reporte Demográfico de usuarios

<b>/api/promo_stats/{:id}/demographic</b>	<b>GET</b>	<b>POST, PUT, DELETE</b>
<p>Header: Authorization: token</p> <p>Body: null</p> <p>id: es el identificador de la promoción sobre la cual se solicita el reporte</p> <p>Query Params:  country (ISO3166-1 largo 2)  city (IATA largo 3)  age (values: 18-25, 25-40, 40-60, 60+)</p>	<p><b>(200)</b> - retorna el reporte demográfico de la promoción cuyo código se recibe como parte de la uri, y aplicando los filtros establecidos con los query params.</p> <p><b>(400)</b> - Bad Request, no se encuentra promoción en el sistema cuyo id concuerde con el especificado en la uri</p>	<p><b>(404)</b> - Not Found</p>

	<p><b>(401)</b> - Unauthorized, el request llega sin token o el token es incorrecto</p> <p><b>(403)</b> - Forbidden, el código de la promoción con id igual al especificado en la uri, no figura dentro de los códigos que el usuario puede consultar y que figuran en el JWT</p>	
--	---	--

## 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-web.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.

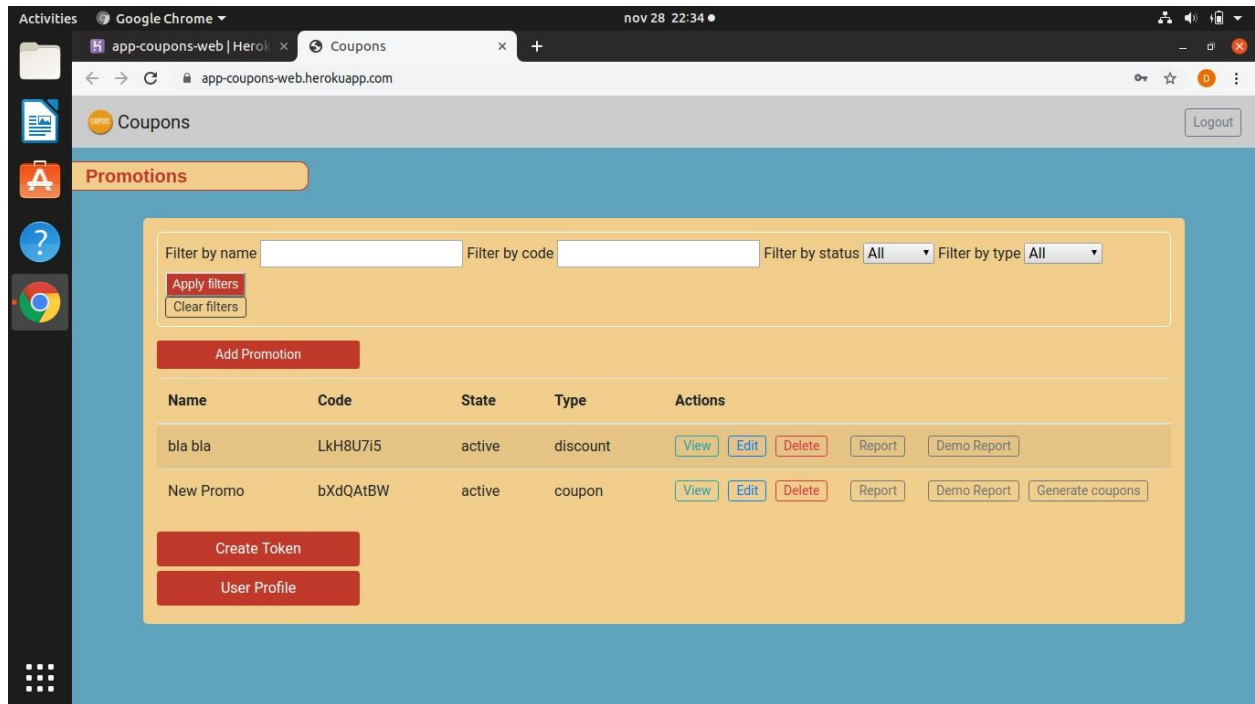


The screenshot shows a web application interface for creating a new user. At the top, there is a navigation bar with the 'Coupons' logo. Below it, a 'New User' tab is highlighted. The main content area features a registration form with the following fields: 'Organization Name', 'Profile picture' (with a 'Choose File' button and 'No file chosen' text), 'Name', 'Last Name', 'Email', 'Password', and 'Password Confirmation'. At the bottom of the form are two buttons: 'Create User' and '<< Back'.

Sign up: <https://app-coupons-web.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.

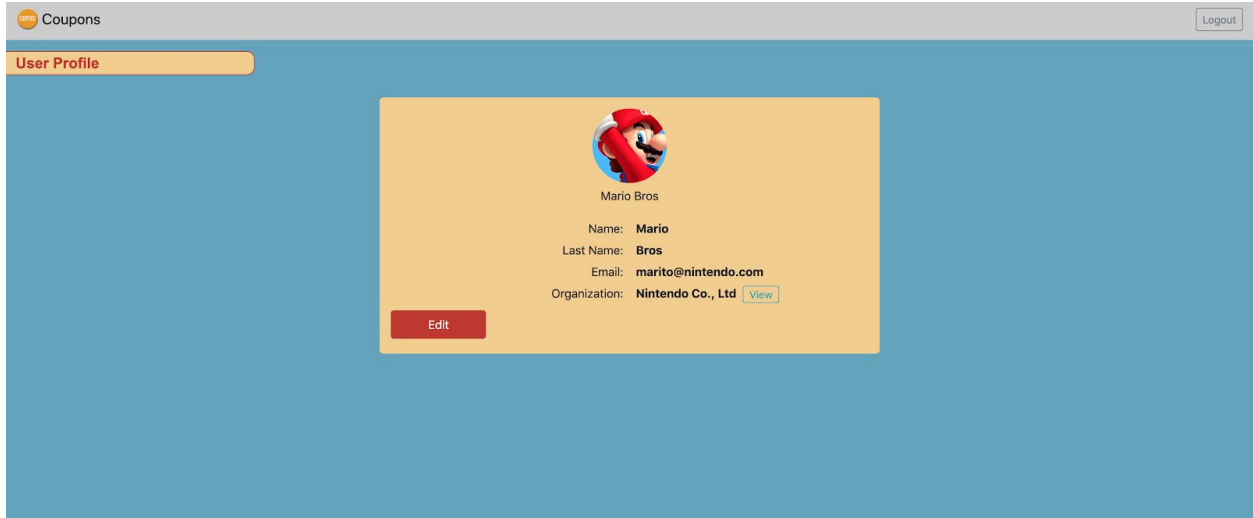


Main Page: <https://app-coupons-web.herokuapp.com>

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.

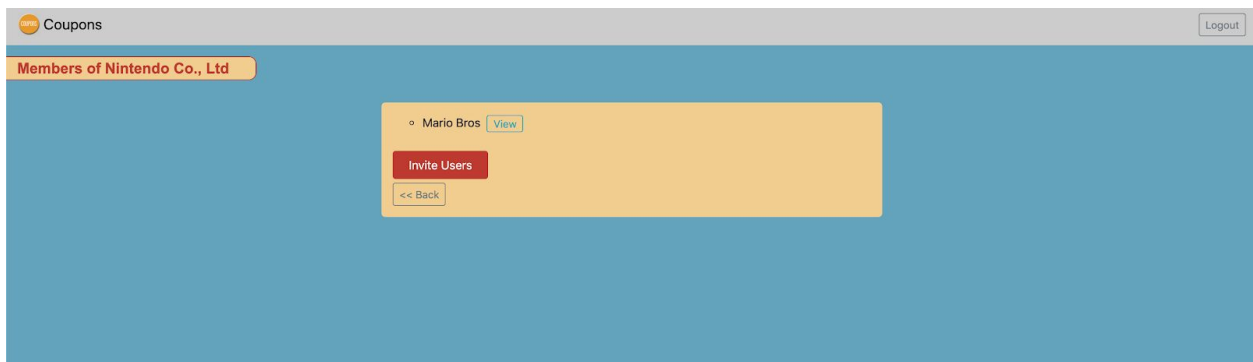
En esta página se visualizan todas las promociones que tienen definidas hasta el momento la organización (en este caso una sola, la promo Navidad), sobre las cuales se pueden realizar un conjunto de acciones (View, Edit, Delete, Report, Generate coupons). Además se puede generar una nueva clave de aplicación 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.



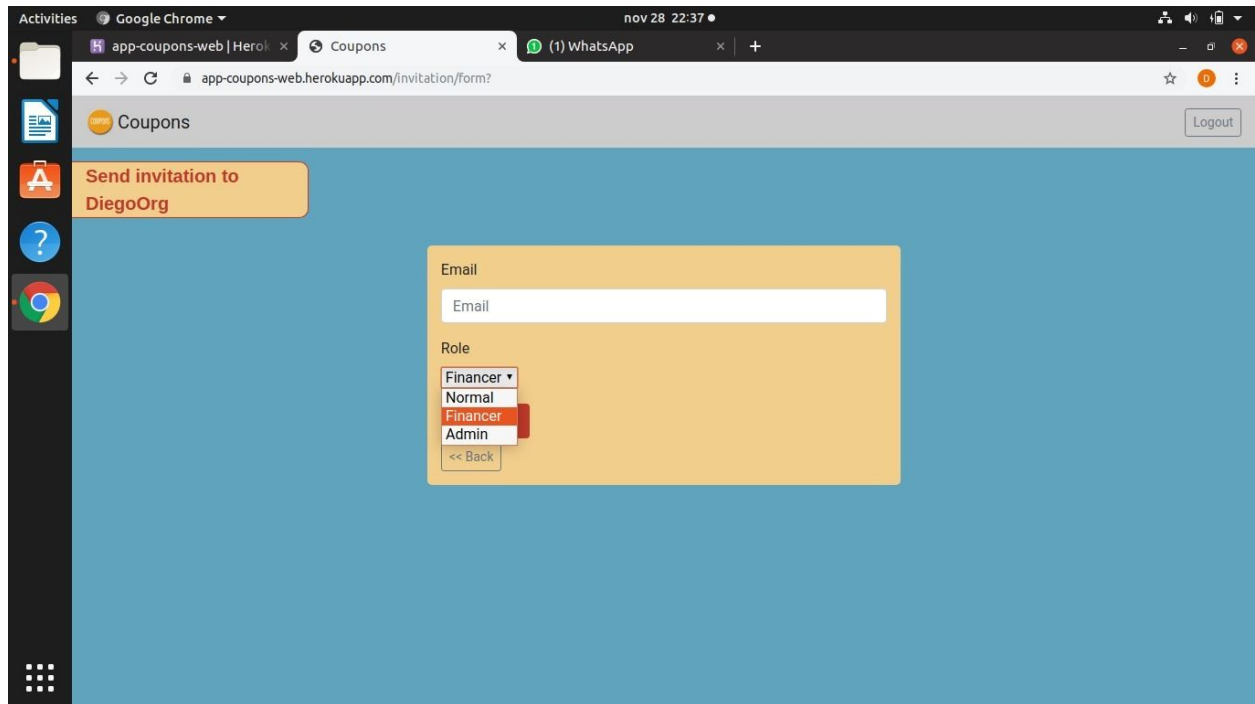
User Profile: <https://app-coupons-web.herokuapp.com/users/{id}>

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.



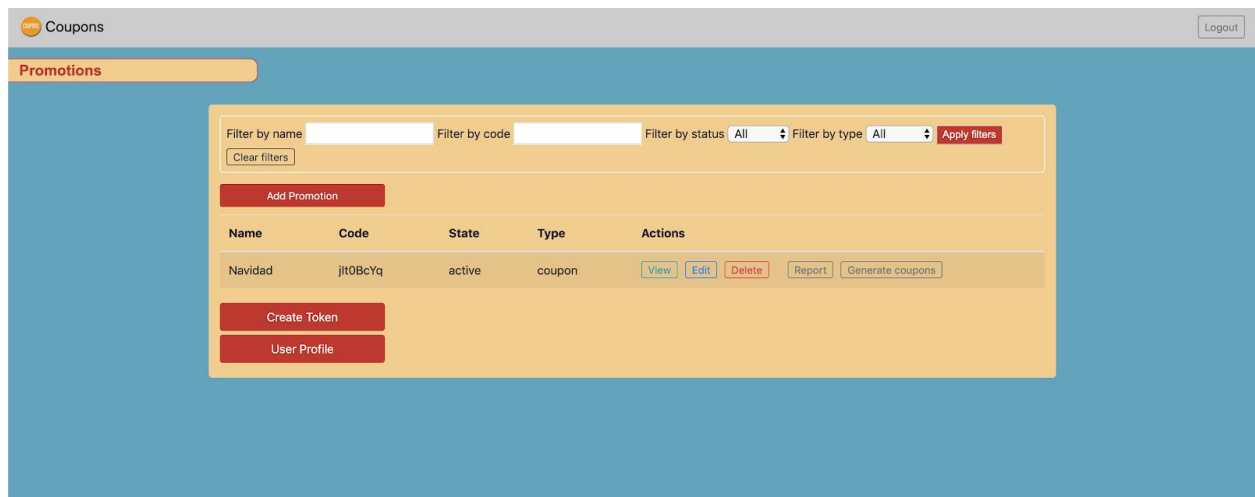
Organization members: <https://app-coupons-web.herokuapp.com/organizations/{id}>

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-web.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-web.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.

New Promotion: [https://app-coupons-web.herokuapp.com/promotion\\_definitions/new](https://app-coupons-web.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.

Name	Code	State	Type	Actions
Carnaval	rHdW63Ah	active	coupon	View Edit Delete Report Generate coupons
Navidad	jlt0BcYq	active	coupon	View Edit Delete Report Generate coupons

Main Page: <https://app-coupons-web.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 browser window with the URL `app-coupons-web.herokuapp.com/promotion_definitions/1/promotions/new?`. The page has a blue background and a sidebar on the left with icons for home, help, and browser. A yellow button labeled "Generate coupons for New Promo" is in the top left. In the center, there is a yellow form with the following fields: "Coupons count" (empty), "Coupons uses" (empty), and "Expiration date" (containing "11/dd/2019"). Below these fields are two buttons: a red "Generate" button and a yellow "<< Back" button. A "Logout" button is in the top right corner.

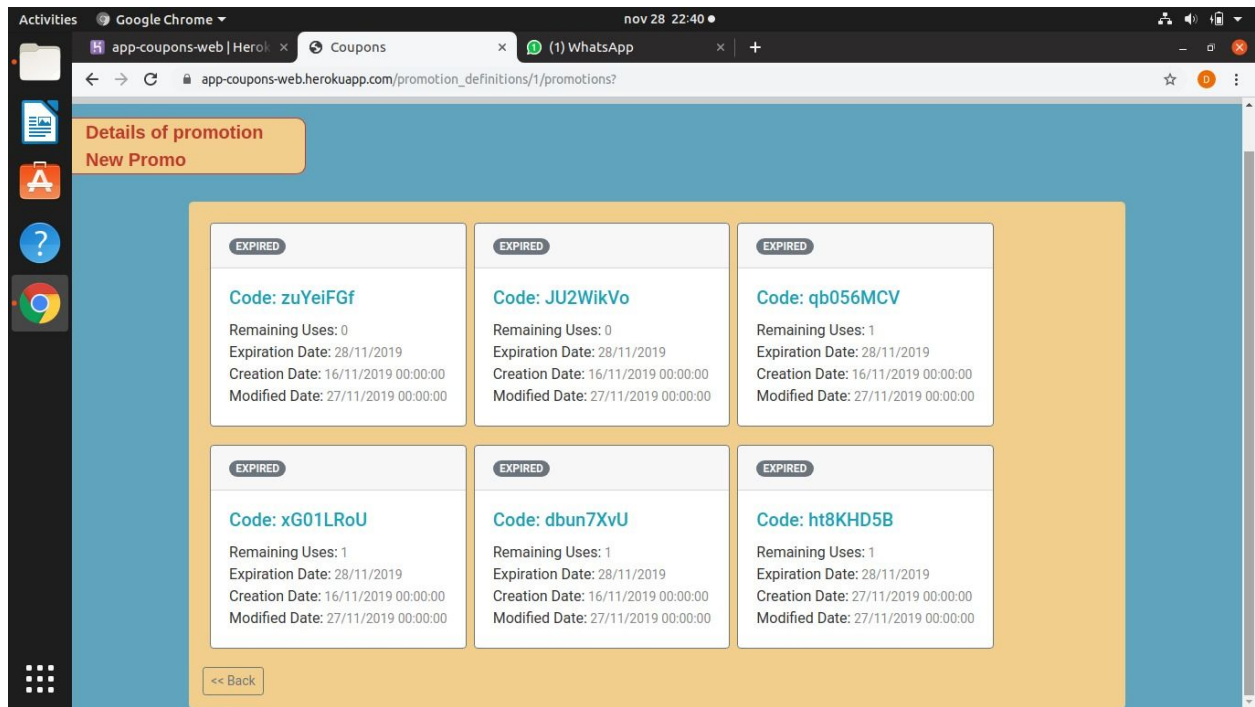
Generate coupons: [https://app-coupons-web.herokuapp.com/promotion\\_definitions/{id}/promotions/new](https://app-coupons-web.herokuapp.com/promotion_definitions/{id}/promotions/new)

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 the "Promotion Carnaval" details page. The page has a blue background and a sidebar on the left with icons for home, help, and browser. A yellow button labeled "Promotion Carnaval" is in the top left. In the center, there is a yellow form displaying the following details: Name: Carnaval, Code: rHdW63Ah, State: active, Type: coupon, Value: 20, Discount type: percentage, Attributes: coupon\_code, total, Conditions: total > 10000. Below these details are two buttons: a red "Show details" button and a yellow "<< Back" button. A "Logout" button is in the top right corner.

Promotion: [https://app-coupons-web.herokuapp.com/promotion\\_definitions/{id}](https://app-coupons-web.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-web.herokuapp.com/promotion\\_definitions/{id}/promotions](https://app-coupons-web.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-web.herokuapp.com/promotion\\_definitions/{id}/report](https://app-coupons-web.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.

Coupons

Promotions

Filter by name: Filter by code: Filter by status: All Filter by type: All Apply filters

Clear filters

Add Promotion

Name	Code	State	Type	Actions
Carnaval	rHdW63Ah	active	coupon	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Report</a> <a href="#">Generate coupons</a>
Navidad	jIt08cYq	active	coupon	<a href="#">View</a> <a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Report</a> <a href="#">Generate coupons</a>

Create Token

User Profile

Main Page: <https://app-coupons-web.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-web.herokuapp.com/promotions/generate\\_token](https://app-coupons-web.herokuapp.com/promotions/generate_token)

Para generar un nuevo token de aplicación, una vez en esta vista debemos ingresar un nombre, seleccionar el o los 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.

Main Page: [https://app-coupons-web.herokuapp.com/promotion\\_definition/{id}/demographic\\_report](https://app-coupons-web.herokuapp.com/promotion_definition/{id}/demographic_report)