Ciencias de la Computación
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS UNIVERSIDAD DE CHILE

Informe de Práctica

CC4901 – Práctica Profesional I

Empresa:	NIC Chile Research Labs
Alumno:	Manuel Olguín
Carrera:	Ingeniería Civil en Computación
RUT:	18.274.982 – 6
E-Mail:	molguin@dcc.uchile.cl
Tel:	+56 9 7463 6997

2 de marzo de 2016
Santiago, Chile.

1. Certificado de la Empresa

2. Observaciones

Índice

1. Certificado de la Empresa **2**

2. Observaciones **3**

3. Resumen **5**

4. Introducción **6**
4.1. Lugar de Trabajo 6
4.2. Equipo de Trabajo 6
4.3. Software y Conceptos Importantes 6
4.3.1. REST 6
4.3.2. API 6
4.3.3. PostgreSQL 7
4.3.4. Python 7
4.3.5. JSON 7
4.3.6. Redis 7
4.3.7. Docker 7
4.3.8. BeCity 7
4.3.9. SUR - Southern Urban Observatory 7

5. Trabajo Realizado **8**

6. Conclusiones **9**

Índice de figuras

3. Resumen

El trabajo se realizó entre Agosto y Octubre del 2015 en NIC Chile Research Labs, el laboratorio de investigación y desarrollo en protocolos de internet de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. Consistió en

1. rediseñar e actualizar la API del backend de BeCity, una aplicación Android desarrollada en el laboratorio, para que se adhiriera al modelo REST.

2. diseñar e implementar un microservicio de manejo de imágenes para el proyecto SUR.

BeCity es una aplicación móvil Android orientada a ciclistas, la cual pretende facilitar el tránsito del usuario por la ciudad. Cuenta con un servicio de búsqueda de rutas con ciclovías, y permite grabar los recorridos del usuario, guardando estadísticas de velocidad, distancia recorrida, calorías quemadas, etc. Funciona bajo el paradigna cliente-servidor: existe un servicio central (el backend) que almacena y maneja los datos, y la aplicación Android (el cliente) se comunica con este servicio para actualizar y obtener información. Esta comunicación se realiza mediante una API (Application Programming Interface) usando HTTP (HyperText Transfer Protocol), sobre la cual se realizó el trabajo previamente mencionado.

El rediseño de esta API tenía como fin su modernización y adaptación a estándares modernos, en específico el modelo REST (REpresentational State Transfer)

SUR (Southern URban Observatory) es una plataforma que pretende centralizar todos los proyectos desarrollados en NICLabs. En este sentido, SUR se plantea como un backend unificado para los distintos servicios, ofreciendo funcionalidades comunes como autentificación de usuarios y manejo de recursos como imágenes, y generando una integración más estrecha entre los servicios.

* explicar microservicio *

4. Introducción

4.1. Lugar de Trabajo

4.2. Equipo de Trabajo

4.3. Software y Conceptos Importantes

* breve introducción *

4.3.1. REST

El modelo REST (por sus siglas en inglés, "Representational State Transfer") es una serie de restricciones arquitecturales que, al aplicarse a un servicio web, induce una serie de propiedades deseables (e.g. escalabilidad, estabilidad, rendimiento, etc). Al estar relacionado sólamente con la estructura arquitectural del software, no especifica detalles de implementación, y es independiente del lenguage de programación escogido.

A continuación, se expondrán brevemente las restricciones impuestas por el modelo REST:

1. **Modelo Cliente - Servidor** Debe existir una clara separación de responsabilidades mediante una interfaz uniforme. Por ejemplo, el servidor debe encargarse del almacenamiento y procesamiento de los datos, pero no de la representación visual de éstos, y vice-versa en el caso del cliente.

2. **Statelessness - Ausencia de Estados** El servidor no debe guardar información del estado del cliente entre solicitudes. El estado es almacenado por el cliente; cada solicitud debe ser independiente y contener toda la información necesaria para realizarse.

3. **Caché** El servidor debe poder guardar solicitudes en caché para poder responder futuras solicitudes de manera más expedita.

4. **Transparencia de Capas** El cliente no debe poder determinar si está conectado directamente al servidor o a un servicio intermedio (proxy, etc).

5. **Uniformidad de la Interfaz** A su vez puede separarse en:

   - *Separación de recursos y su representación* La representación externa de los recursos (datos) debe ser independiente de como son almacenados por el servidor. A su vez, estas representaciones contienen toda la información necesaria para poder modificar el recurso original.
   - *Identificación Uniforme de Recursos* Por ejemplo, mediante URI's[1].
   - *Mensajes Autodescriptivos* Cada mensaje incluye información de cómo procesarlo.

4.3.2. API

Una API (Application Programming Interface), en el contexto de servicios web, es una interfaz de interacción entre los componentes de un modelo servidor-cliente. Específicamente, consiste en un conjunto de puntos de enlace (endpoints) públicos en el servidor, a través de los cuales un cliente puede comunicarse e interactuar con el servicio central.

4.3.3. PostgreSQL

4.3.4. Python

Python es un lenguaje de programación interpretado, dinámico y de alto nivel, con énfasis en la fácil lectura de su código y su brevedad sintáctica. Es un lenguaje multipropósito, apto para programas de pequeña y gran escala, y con soporte para múltiples paradigmas de programación (orientado a objetos, funcional, imperativo, etc).

En el contexto del trabajo realizado, Python, principalmente en su versión 2.7 (aunque en algunas instancias se ocupó también la versión más nueva del lenguaje, 3.5), fue el lenguaje escogido para el desarrollo de los servicios. Esto, ya que debido a su gran popularidad, cuenta con una enorme cantidad de librerías bien documentadas y mantenidas que facilitaron el desarrollo. Principalmente, se utilizaron las librerías

- *Flask*[2], la cual provee funcionalidades para el desarrollo de servicios web.
- *Restless*[3], para implementar el modelo REST.
- *SQLAlchemy*[4], para interactuar con la base de datos.

4.3.5. JSON

JSON (JavaScript Object Notation), es un estándar abierto de codificación de datos para la comunicación entre servidor y cliente, en la cual los datos se condifican en pares atributo-valor en "cleartext" (es decir, texto legible por humanos). Por ejemplo, el siguiente extracto ilustra una posible codificación de la FCFM en JSON:

```
{
    "Nombre": "Facultad de Ciencias Fisicas y Matematicas",
    "Universidad": "Universidad de Chile",
    "Direccion": {
        "Calle": "Beauchef",
        "Numero": 850,
        "Comuna": "Santiago",
        "Region": "Metropolitana",
        "Pais": "CHILE"
    }
}
```

4.3.6. Redis

4.3.7. Docker

4.3.8. BeCity

4.3.9. SUR - Southern Urban Observatory

5. Trabajo Realizado

6. Conclusiones

Referencias

[1] *URI - Uniform Resource Identifier*: cadena de caracteres utilizada para identificar recursos específicos, siguiendo un esquema definido.
`https://en.wikipedia.org/wiki/Uniform_Resource_Identifier`

[2] *Flask is a microframework for Python based on Werkzeug, Jinja 2 and good intentions.*
`http://flask.pocoo.org/`

[3] *Restless - A lightweight REST miniframework for Python.*
`https://restless.readthedocs.org/en/latest/`

[4] *SQLAlchemy - The Python SQL Toolkit and Object Relational Mapper*
`http://www.sqlalchemy.org/`