Informe de Trabajo Profesional

"Stockizer"

Motor de procesamiento de instrumentos bursátiles para utilización de trading algorithms

Autores

L. Simonelli, Padrón 93111, lsimonelli@fi.uba.ar
T. Boccardo, Padrón 93637, tboccardo@fi.uba.ar
J. Stankus, Padrón 93143, jstankus@fi.uba.ar

Carrera

Ingeniería en Informática

Director

Lic. Pablo Gustavo Cosso

Índice

Objetivo del proyecto

El objetivo del presente proyecto es el desarrollo de una aplicación que brinde información de cotizaciones de instrumentos financieros que se negocien en cualquier ámbito bursátil, permitiendo el uso de modelos cuantitativos de decisión para determinar las decisiones de compra y venta de los distintos activos disponibles.

Visión del proyecto

Con el desarrollo de este proyecto se pretende lograr a futuro una mayor incorporación de la tecnología en el mercado bursátil Argentino. Hoy en día, éste se encuentra muy atrasado en comparación con otros mercados de la región de América Latina como Chile o Brasil. Se busca contribuir a la madurez del mismo y colaborar con el crecimiento de la tendencia en la utilización de software en la operación bursátil Argentina.

Análisis de Mercado y motivación

En el mercado bursátil existen inversores que se dedican a realizar compras utilizando el concepto de modelos cuantitativos. Es decir, construyen modelos matemáticos y en base a éstos determinan operaciones de compra y venta de manera tal que se pueda obtener el máximo rédito posible. Esta forma de actuar se denomina 'Algorithmic trading'[1]. En el año 2012, un estudio dio como resultado que el 50% del trading en EE UU se realizó con este método[2]. Esta práctica es incipiente en América Latina y en particular en Argentina y es vital para el negocio que lleva a cabo nuestro cliente, Francisco Prack.

En nuestro país, los parámetros de entrada de estos modelos se pueden obtener de la aplicación BOLSAR[3]. Esta aplicación, escrita en Visual Basic, lleva ya varios años sin actualizaciones. Se le aplican los modelos a estos datos y se obtiene la decisión de operar, que luego debe ser transmitida a un agente de bolsa para que realice la operación.

Estos factores agregan demoras al proceso. Tratándose de operaciones en las que se puede dar diferencia entre beneficio y pérdidas por un par de minutos, las demoras constituyen un factor crítico.

Por lo tanto, se nos acercó la necesidad de optimizar estos procesos, desarrollando una aplicación que permita el acceso a los datos en tiempo real o, de no ser posible por limitaciones de la fuente de datos, servirlos con la mayor frecuencia posible, automatizando también el proceso de carga de reglas y notificación de operaciones.

Adicionalmente, desde la bolsa de comercio se planea integrar en el transcurso del siguiente año, el sistema Millenium[4], de la bolsa de Londres. Este sistema implica varias mejoras de performance respecto del sistema actual, con lo que se buscará abstraer la entrada de la aplicación a desarrollar para poder integrarlo en el futuro.

---

[1] https://en.wikipedia.org/wiki/Algorithmic_trading
[2] http://topics.nytimes.com/top/reference/timestopics/subjects/h/high_frequency_algorithmic_trading/
[3] https://www.bolsar.com/VistasDL/PaginaPrincipal.aspx
[4] http://goo.gl/GTm0un

Propuesta de solución

Roles del sistema

- Usuario: persona que accede a los datos generados por la aplicación. Puede pedir reportes, gráficos, ingresar reglas, configurar la compra.
- Administrador: configura los tiempos de ejecución de las tareas periódicas, crea los usuarios y configura los inputs.
- Agente: recibe una notificación cuando debe comprar o vender los bonos.

Requerimientos funcionales

1. **Módulo de procesamiento de datos**
    a. Lectura de datos
    b. Ejecución periódica configurable.
    c. Procesamiento de los datos
    d. Filtrado de datos
    e. Normalización de los datos
    f. Almacenamiento de datos normalizados
    g. Creación de tarea de decisión.
2. **Módulo de decisión**
    a. Ejecución de script de decisión
    b. Output
        i. Output a base de datos de los resultados
        ii. Output de info relevante estadísticamente
3. **Módulo de salida.**
    a. Notificaciones a agentes de bolsa
        i. Mail
        ii. Skype
    b. Registro de transacciones, resultados y notificaciones
4. **API**
    a. Servicio de datos de valores (en vivo y por periodo de tiempo)
    b. Servicio de generación de reportes
    c. Servicio de generación de gráficos
    d. Servicio de datos sobre movimientos realizados
    e. Servicio para configurar frecuencia de refresco de datos (admin)
    f. Servicio para configurar notificaciones de salida (admin)
5. **Web frontend**
    a. Visualización de datos en vivo.
    b. Interfaz para exportar reportes a PDF y CSV.
    c. Interfaz para visualización de gráficos
    d. Interfaz para visualizar operaciones realizadas
6. **Web backoffice**
    a. Interfaz web para configurar periodo de recopilación.
    b. Interfaz web para configurar notificaciones de salida

Requerimientos no funcionales

UI/UX - Portabilidad

- Responsive Design
- HTML5/CSS3

Seguridad

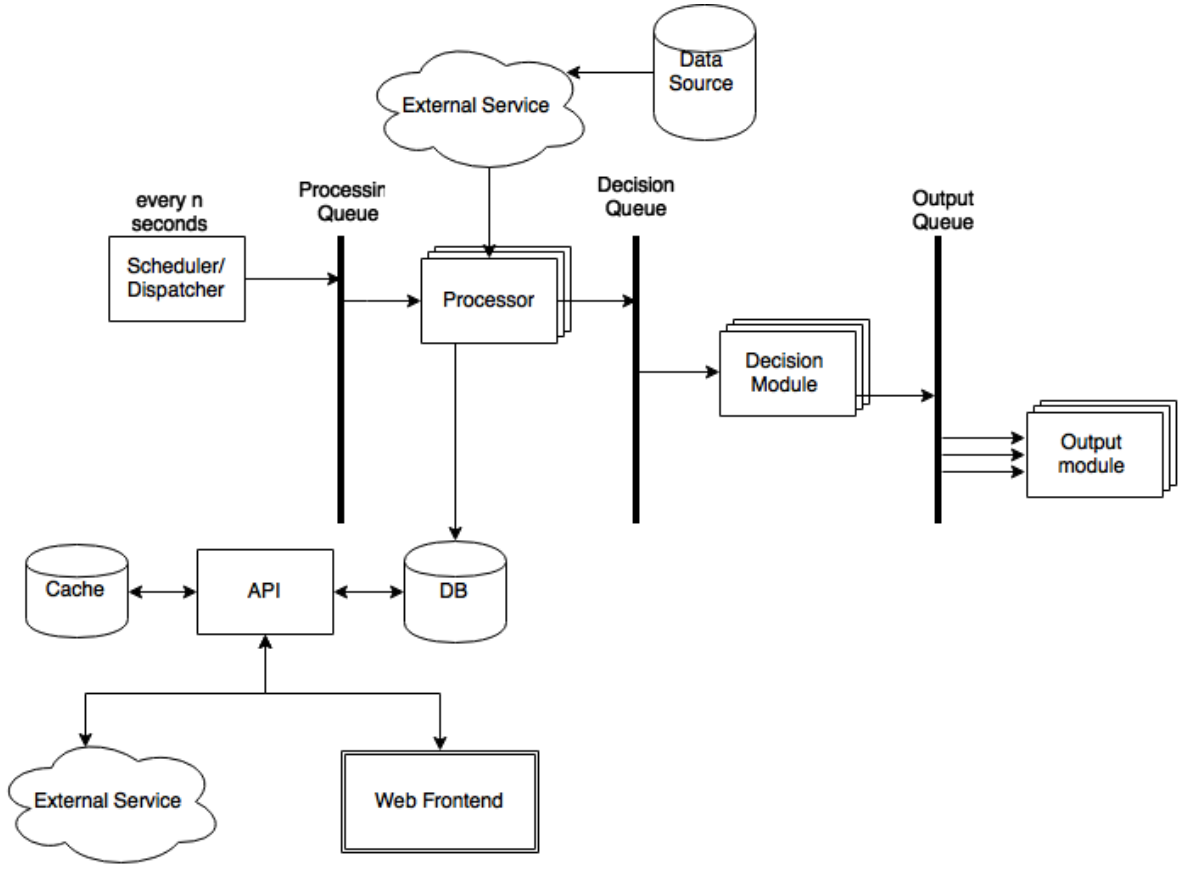
- API segura implementando algún protocolo de autenticación (OAuth, token auth, etc)
- Protección contra inyección SQL, XSRF, y demás ataques del top 10 de OWASP[5]

Calidad

- Real time.
- Unit Testing.
- Pruebas de Integración.

Tecnología

Arquitectura de Alto Nivel



La solución propuesta presenta una arquitectura basada en el patrón de MicroServicios.

El patrón de MicroServicios consiste en separar responsabilidades en distintos módulos independientes comunicados de manera sincrónica usando requests HTTP/REST o

[5] https://www.owasp.org/index.php/Top_10_2013-Top_10

asincrónica utilizando el protocolo AMQP. En nuestro caso, separamos en tres módulos de procesamiento de información comunicados mediante RabbitMQ. El primer módulo es responsable de descargar la información de la fuente de datos, procesarla, normalizarla y guardarla en la base de datos. El segundo módulo se encarga de analizar los datos de los valores normalizados y tomar decisiones sobre qué operaciones se deben realizar. El último módulo se encarga de realizar las operaciones que surgieron del módulo de decisión y notificar a los interesados.

Además se realizará una separación entre el front end y la api, que tendrá la responsabilidad de servir a éste y a los servicios externos que la consuman. El módulo de la api tendrá acceso a la base de datos y a una caché donde serán almacenados los datos más usados para mejorar la performance de las solicitudes más comunes.

Lenguaje y plataformas

Para todos los procesos que realizan las tareas de Scheduling, Procesamiento, Decisión y Output, y para la API, se utilizará Python. Python[6] es un lenguaje de tipado dinámico e interpretado que es fácil de usar y mantener, lo que ayuda a disminuir el esfuerzo de desarrollo en gran medida. Sus librerías son de código abierto y tienen soporte constante de la comunidad open source.

En el desarrollo de la aplicación cliente que consumirá la API principal, se utilizará JavaScript (ECMAScript 6), ya que es el lenguaje que soportan los browsers hoy en dia.

Base de datos

Dado que el modelo de datos no será propenso a variar en gran medida durante el ciclo de vida del producto, se eligió utilizar una base de datos relacional, optando por PostgreSQL.

Cache

Para la implementación de una política de caching en la API, se utilizará Redis[7], una base de datos en memoria que cuenta con soporte para distintos tipos de estructuras de datos.

Cola de mensajes

Para la comunicación entre los distintos procesos del sistema, se optó por el uso de colas de mensajes. Una opción muy popular en este tipo de soluciones es RabbitMQ[8], que es open source y está disponible como servicio de forma online (utilizando un servidor externo)[9].

Dependencias y herramientas

- Frameworks
    - Celery (Framework para cola de tareas distribuidas para Python)
    - Flask Restful (Framework de API REST para Python)
    - SQLAlchemy (ORM para Python)
    - Angular.js (MVC Framework para Javascript)
- Ambiente de desarrollo y producción

---

[6] https://www.python.org/
[7] http://redis.io/
[8] https://www.rabbitmq.com/features.html
[9] Por ejemplo, https://www.cloudamqp.com/

- Docker

Estimación de Esfuerzo

La estimación fue realizada mediante las técnicas *planning poker*[10] (de Scrum) y *bottom up*[11]:

Esfuerzo de Desarrollo

ID	Descripción	Esfuerzo (hs)
1	**Módulo de procesamiento**	
1.1	Scheduler de ingestión	8
1.2	Mocks de data source externo	8
1.3	Obtención de información de la bolsa de Buenos Aires	50
1.4	**Procesamiento de datos**	
1.4.1	Lectura de datos	2
1.4.2	Normalización de datos	8
1.5	Creación y scheduling de tareas de decisión	2
3	**Módulo de decisión**	
3.1	Integración con funciones de filtrado existentes en VBA	65
3.2	Carga y ejecución de reglas de decisión	85
3.4	Creación y scheduling de tarea de output	2
4	**Módulo de salida**	
4.1	Registro de transacciones	5
4.2	Notificaciones	18
4.3	Registro de notificaciones	5
5	**API**	
5.1	Proof of Concept de una API básica en Go	6

[10] http://es.wikipedia.org/wiki/Planning_poker
[11] http://4pm.com/bottom-up-estimating/

5.2	Seguridad y autenticación	40
5.3	Servicio de datos de valores	34
5.4	Servicio de datos sobre movimientos realizados	34
5.5	Servicio para configurar frecuencia de refresco de datos (admin)	34
5.6	Servicio para configurar notificaciones de salida (admin)	34
5.7	Servicio para alta de instrumentos financieros (admin)	25
5.7	Servicio para generación de reportes	44
5.8	Servicio de carga de script de decisión	23
6	**Web frontend**	
6.1	Interfaz de login y registro	40
6.2	Interfaz de generación de reportes en formato CSV y PDF	13
6.3	Visualización de datos en vivo	13
6.4	Visualización de operaciones realizadas	13
7	**Web back office**	
7.1	Interfaz web para configurar periodo de recopilación	13
7.2	Interfaz web para configurar notificaciones de salida	34
7.3	Interfaz web para alta de instrumentos financieros	13
	TOTAL	**671**

Esfuerzo del proyecto

ID	Descripción	Esfuerzo (hh)
1	**Análisis de requerimientos**	
1.1	Identificación de requerimientos funcionales y no funcionales	21
1.2	Análisis de requerimientos	40
1.3	Estimación requerimientos	13
2	**Configuración de ambientes**	
2.1	Configuración de repositorio de versionado	1
2.2	Configuración de servidores de producción y testing	13
2.3	Configuración de entorno de desarrollo	8
2.4	Configuración de base de datos	8
3	**Diseño**	
3.1	Diseño de arquitectura	13
3.2	Diseño de modelo de dominio	13
3.3	Diseño de modelo de persistencia	13
3.4	Diseño de interfaz de usuario	41
4	**Desarrollo**	**671**
5	**Testing**	
5.1	Testing automático	
5.1.1	Modelo de dominio	13
5.1.2	Módulo de Procesamiento	26
5.1.3	Módulo de Decisión	34
5.1.4	Módulo de Output	13
5.1.5	API	21
5.2	Stress testing	13
5.3	Definición y ejecución de casos de prueba manuales de UI	55

6	**Documentación**	
6.1	Generación de documentación técnica	32
6.2	Redacción de un manual de usuario	13
7	**Gestión**	
7.1	Gestión de requerimientos, alcance y cambios	21
7.2	Gestión de riesgos	21
7.3	Planificación y calendarización	55
7.4	Seguimiento de avance	34
7.5	Gestión de entregas incrementales	34
7.6	Gestión de la calidad del producto	21
8	**Presentación**	42
	TOTAL	**1303**

Plan de trabajo

Se estima que el trabajo completo a realizar tomará 6 meses, con una dedicación de 71 hs mensuales al desarrollo del trabajo de parte de cada integrante. El proceso de desarrollo será iterativo e incremental, con 6 iteraciones de 1 mes, a lo largo de las cuales se efectuarán tanto tareas de análisis, desarrollo, testing y documentación como tareas de gestión, cada una con distinta carga horaria dependiendo de la etapa del proyecto.

Para la gestión del proyecto, se adoptarán algunas prácticas de metodologías ágiles. Como en Scrum, se realizarán reuniones de Planificación, Revisión y Retrospectiva para cada iteración y una reunión informal diaria de reporte de estado que se llevará a cabo tanto presencialmente como de forma remota dependiendo de la disponibilidad de los involucrados. Se utilizará un tablero de Kanban[12] para organizar el trabajo en cada iteración.

Se efectuará la gestión de riesgos con una planilla en donde se registrarán los riesgos identificados, junto con su probabilidad de manifestación, impacto sobre el objetivo del proyecto y su exposición estimada. Se describirán también los planes de respuesta y planes de contingencia para cada riesgo.

Se buscará que el contacto con el tutor y el cliente se efectúen principalmente durante las reuniones de Revisión, aunque se intentará integrar al cliente al proceso de desarrollo día a día para facilitar la comunicación de requerimientos y evitar el retrabajo. Durante las reuniones de Revisión, se presentarán tanto al tutor como al cliente el avance alcanzado durante la iteración anterior. Se definirán prioridades, nuevos requerimientos y cambios en la planificación.

A continuación se presenta la planificación de cada iteración. Es importante destacar que las estimaciones iniciales de esfuerzo pueden perder validez a medida que avanza el proyecto, es por esto que se destinará tiempo para realizar nuevas estimaciones, utilizando los métodos *planning poker* y *bottom up* mencionados previamente:

Iteración 1

- Confección de Plan de Proyecto.
- Identificación de requerimientos funcionales y no funcionales.
- Análisis de requerimientos
- Estimación de requerimientos
- Diseño de arquitectura.
- Configuración de entorno de desarrollo, base de datos y repositorio de versionado.
- Diseño del modelo de persistencia, modelo de dominio
- Módulo de procesamiento:
    - Scheduler de procesamiento
    - Obtención de información de la bolsa de Buenos Aires
    - Mocks de data source externo

[12] https://en.wikipedia.org/wiki/Kanban_board

- Lectura de datos
- Normalización de datos
- Creación y scheduling de tareas de decisión

POC de API básica en Go

Iteración 2

API:

- Seguridad y Autenticación
- Servicio de datos de valores
- Servicio de configuración de periodos de recopilación de datos
- Servicio para alta de instrumentos financieros

Web Frontend:

- Diseño de interfaz de usuario
- Visualización de datos en vivo

Iteración 3

Módulo de decisión:

- Carga y ejecución de reglas de decisión
- Integración con funciones de filtrado existentes en VBA
- Creación y scheduling de tarea de output

Web frontend:

- Interfaz de login y registro

Iteración 4

Módulo de salida:

- Notificaciones
- Registro de notificaciones
- Registro de transacciones

API:

- Servicio de carga de script de decisión
- Servicio de frecuencia de refresco de datos (admin)
- Servicio para generación de reportes

Iteración 5

API:

- Servicios para notificaciones de salida (admin)
- Servicio de datos sobre movimientos realizados

Web frontend:

- Visualización de operaciones realizadas
- Interfaz de generación de reportes en formato CSV y PDF

Web backoffice:

- Interfaz web para alta de instrumentos financieros
- Interfaz web para configurar periodo de recopilación
- Interfaz web para configurar notificaciones de salida

Iteración 6

- Generación de documentación técnica
- Redacción de manual de usuario
- Definición y ejecución de casos de prueba manuales de UI
- Stress testing
- Hacer presentación
- Ensayar presentación

Resultados

Se implementó el sistema, en líneas generales, según lo planeado y en el tiempo estipulado. Se excluyeron algunas tareas del alcance inicial y se incluyeron otras.

Se recortó del alcance la generación de reportes en formato PDF y el registro de usuarios nuevos, debido a algunas adiciones que fueron necesarias.

En el módulo de decisiones, no se utilizaron scripts VBA, como se había planeado inicialmente, sino que se codificaron scripts de carga escritos en el lenguaje de programación Python. Este cambio se acordó con el cliente, debido a una necesidad de simplificar este proceso.

Otro cambio que se tuvo que afrontar fue la implementación de varios procesadores de datos de la Bolsa de Comercio de Buenos Aires. Se comenzó implementando un procesador que consumía Global Equity, pero desafortunadamente se perdió el acceso a este servicio y se debió implementar un procesador nuevo para el servicio de PuenteNet.

Con el objetivo de realizar pruebas y para la demostración final, se creó un módulo de procesamiento con datos aleatorios, que permite correr la aplicación en horarios en que la Bolsa Comercio de Buenos Aires no está en operación.

Problemas encontrados durante el desarrollo

- Se realizó una prueba de concepto para implementar un módulo de procesamiento de Bolsar, pero no se obtuvieron resultados positivos, por lo que se descartó es opción.
- Durante el desarrollo, el cliente no pudo participar más, por lo que el trabajo continuó sin su colaboración.
- Se implementó un módulo de procesamiento de Global Equity, que luego no se pudo utilizar, ya que se necesitaba que el cliente diera acceso al sitio.
- Muchas de las pruebas se realizaron cuando el mercado bursátil no operaba, por lo que se hicieron desarrollos ad-hoc con el propósito de mostrar la operación del sistema. Estos desarrollos se utilizan también para alimentar al sistema en demostraciones del mismo.

Planes a futuro

- Soportar varios scripts de decisión para poder comparar sus predicciones sobre un mismo set de cambios en los valores.
- Aprendizaje automático de las decisiones: las decisiones pueden tener “memoria” e ir aprendiendo en base a experiencia pasada, en función de si lograron predecir correctamente o no.
- Soportar múltiples usuarios y monetizar la solución, probablemente bajo la modalidad SaaS.
- API pública.
- Integrar con Millenium para lograr mayor velocidad y automatización.
- Incluir distintos tipos de instrumentos financieros.
- Incluir más proveedores de distintas bolsas.

Glosario

Sistema de entrada: Es el Sistema externo que proveerá los datos asociados a los Bonos sobre los cuales trabajará el sistema a implementar.
Bono: Un Bono es un instrumento financiero de deuda que sirve para financiar a las entidades que lo emiten, teniendo como ejemplo notable al estado. Llegada cierta fecha límite, los Bonos se pagan con cierta tasa de interés. Los distintos datos asociados al bono (precio, intereses, fecha de pago, etc) son los que requiere el Modelo de decisión que notificará la compra que se deberá efectuar.
Modelo de decisión: Un Modelo de decisión, dentro del sistema a implementar, es el que, en base a la información obtenida sobre los distintos Bonos disponibles de cierto sistema de entrada (BOLSAR, Millenium, etc), decide que movimientos de compra y venta de Bonos deben efectuarse.

Currículum de los autores

Tomás Boccardo

Información Personal

DNI: 36.275.679
Fecha de Nacimiento: 23 de Julio de 1991
Estado Civil: Soltero
Nacionalidad: Argentina
Teléfono: +54 9 11 6352 6506
E-mail: tomasboccardo@gmail.com

Intereses

Backend Development. Python. Node. Open Source. Docker. Linux.

Educación

Facultad de Ingeniería de la UBA, Ciudad Autónoma de Buenos Aires

Ingeniería en Informática. *Esperado: 2016*

- Actualmente cursando el último cuatrimestre de la carrera.
- Promedio: 7

Instituto San Luis, San Fernando, Buenos Aires

Título Secundario, *2009*

- Bachiller con orientación en ciencias naturales
- Promedio: 8

Idiomas

Inglés - Nivel Avanzado

Cambridge Certificate in Advanced English

- Año 2009
- Grade C: 71/100

Cambridge First Certificate in English

- Año 2007
- Grade B

Experiencia Laboral

Full Stack Web Developer en FDVSolutions *Febrero 2014 - actualidad*

- Backend development using Node.js, Django, Flask, Grails, Ruby on Rails.
- Frontend development for web and mobile using HTML5, CSS3 y JavaScript with frameworks like Angular, Backbone, Require and Phonegap.
- Database management with MySQL, PostgreSQL, MongoDB.
- Linux System Administration

Soporte Técnico en Tecna *Septiembre 2011 - Febrero 2014*

- Soporte y Reparación de PC, manejo de software y hardware
- Soporte y conexión de internos telefónicos.
- Seguimiento de casos, soporte telefónico remoto y on-site de usuarios.
- Manejo de equipos de telefonía
- Seguimiento y soporte de parque de impresoras.

Cursos

Udacity

Interactive 3d Graphics, *Junio 2013*

- Conceptos importantes de computación gráfica
- Especificación WebGL para renderización de 3d en el navegador
- Framework Three.js para desarrollo 3d en JavaScript

Web Development, *Junio 2013*

- Conceptos importantes para el desarrollo de aplicaciones Web
- HTML5, CSS3
- Webapp2 web framework y Jinja2 template engine para Python. Django web framework para Python.
- Google App Engine datastore. PostgreSQL. MySQL.
- Conceptos básicos de criptografía y seguridad aplicado al desarrollo Web.

Conocimientos

Programación

- **Lenguajes:** Python, JavaScript, Ruby, Groovy, Java
- **Frameworks:** Celery, Flask, Django, SQLAlchemy, Express, Underscore, Angular, Backbone, Require, Rails, Grails, Android SDK.
- **Base de Datos:** Postgres, MySQL, MongoDB, Redis.
- **Cola de Mensajes:** RabbitMQ
- **Maquetado:** HTML5, CSS3.
- **Herramientas de Virtualización:** Docker, Vagrant, VirtualBox, VMWare.
- **Herramientas de Despliegue:** Ansible, Heroku, Bash.
- **Sistema Operativo:** Linux

Simonelli Lucas

Lucas Simonelli

Información Personal

DNI: 36.081.648
Fecha de Nacimiento: 10 de Septiembre de 1991
Estado Civil: Soltero
Nacionalidad: Argentina
Teléfono: +54 9 11 5332 0702
E-mail: lucasp.simonelli@gmail.com

Intereses

Backend Development. Python. NodeJS. Rails. Linux.

Educación

Facultad de Ingeniería de la UBA, Ciudad Autónoma de Buenos Aires
Ingeniería en Informática. *Esperado: 2016*

- Actualmente cursando el último cuatrimestre de la carrera.
- Promedio: 7

E.E.T N°2 Alemania, San Martín, Buenos Aires
Título Secundario, *2009*

- Bachiller con orientación en bienes y servicios
- Promedio: 9

Idiomas

Inglés - Nivel Avanzado
Cambridge First Certificate in English

- Año 2010
- Grade A

Experiencia Laboral

Developer en FDVSolutions *Febrero 2014 - actualidad*

- Backend development using Node.js, SpringMVC
- iOS development.
- Database management with MySQL, PostgreSQL, MongoDB.
- Linux & Mac System Administration

Conocimientos **Programación**

- **Lenguajes:** Python, JavaScript, Ruby, Groovy, Java, Objective-C, Swift
- **Frameworks:** Express, Underscore, Angular, Rails, Grails, Cocoa, Spring, Hibernate
- **Base de Datos:** Postgres, MySQL, MongoDB, Redis.
- **Cola de Mensajes:** RabbitMQ, Amazon SQS
- **Herramientas de Virtualización:** Docker, Vagrant, VirtualBox, VMWare.
- **Herramientas de Despliegue:** Heroku, Bash.
- **Sistema Operativo:** Linux, MacOS

Joaquin Stankus

Información Personal

DNI: 36.931.189
Fecha de Nacimiento: 17 de Junio de 1992
Estado Civil: Soltero
Nacionalidad: Argentina
Teléfono: +54 9 11 5752 5905
E-mail: joaquinstan@gmail.com

Educación

Facultad de Ingeniería de la UBA, Ciudad Autónoma de Buenos Aires
Ingeniería en Informática. *Esperado: 2016*

- Actualmente cursando el último cuatrimestre de la carrera.
- Promedio: 7

Belgrano Day School, Belgrano, Ciudad Autónoma de Buenos Aires
Título Secundario, *2009*

- Bachiller bilingüe con orientación en ciencias exactas, físicas y naturales
- Promedio: 7.3

Idiomas

Inglés - Nivel Avanzado
Cambridge A/S Level Language and Literature in English

- Año 2009
- Grade A

Experiencia Laboral

Full Stack Developer en FDVSolutions *Julio 2014 - actualidad*

- Desarrollo backend Node.js (JavaScript), Django (Python)
- Desarrollo frontend web y mobile con HTML5, CSS3 y JavaScript con frameworks como Backbone, Require y Cordova Phonegap.
- Gestión de bases de datos MySQL, PostgreSQL, MongoDB.
- Administración de sistemas Linux

Pasante en Web Development Team en Kimberly-Clark
Octubre 2012 - Noviembre 2013

- Desarrollo Web Full-Stack en ASP .NET MVC 3 y ASP .NET

Conocimientos

Programación

- **Lenguajes:** Python, JavaScript, Ruby, Groovy, Java
- **Frameworks:** Celery, Django, Express, Underscore, Backbone, Require, Rails, Grails, Android SDK.
- **Base de Datos:** Postgres, MySQL, MongoDB, Redis.
- **Cola de Mensajes:** RabbitMQ
- **Maquetado:** HTML5, CSS3.
- **Herramientas de Virtualización:** Docker, VirtualBox, VMWare.
- **Herramientas de Despliegue:** Heroku, Bash.
- **Sistema Operativo:** Linux

Plan de Cursado

Tomás Boccardo (Orientación: Gestión Industrial de Sistemas)

2do Cuatrimestre 2015

[75.47] Taller de Desarrollo de Proyectos II
[75.46] Administración y Control de Proyectos Informáticos II
[75.48] Calidad en Desarrollo de Sistemas
[75.69] Sistemas Automáticos de Diagnóstico y Detección de Fallas II
[75.71] Seminario de Ing. en Informática

Lucas Simonelli (Orientación: Gestión Industrial de Sistemas)

2do Cuatrimestre 2015

[75.47] Taller de Desarrollo de Proyectos II
[75.46] Administración y Control de Proyectos Informáticos II
[75.48] Calidad en Desarrollo de Sistemas
[75.69] Sistemas Automáticos de Diagnóstico y Detección de Fallas II
[75.71] Seminario de Ing. en Informática

Joaquín Stankus (Orientación: Gestión Industrial de Sistemas)

2do Cuatrimestre 2015

[75.47] Taller de Desarrollo de Proyectos II
[75.46] Administración y Control de Proyectos Informáticos II
[75.48] Calidad en Desarrollo de Sistemas
[75.69] Sistemas Automáticos de Diagnóstico y Detección de Fallas II
[78.01] Idioma Inglés