UNIVERSIDAD FASTA

Ingeniería en Informática – FIM 42 – Proyecto Final

**Sprint 0**

# CommonJobs

**(Sistema de Recursos Humanos CommonSense)**


**Alumnos**:

* Andrés Moschini.
* Matías José.
* Juan Diego Raimondi.

**Director Funcional**: Gabriel Buyatti.

**Director Técnico**: Ing. Alejandro Fantini.

**Auditora**: Ing. Ana Haydee Di Iorio.

**Cátedra**:

* Profesor titular: AS. Hilario Fernando Schechtel
* Profesor asociado: Ing. Roberto Giordano Lerena
* Profesor asociado: Lic. Alejandro Nikolic



**Fecha de presentación**: 17/02/2012

# Tabla de Contenidos

Tabla de Contenidos ..... 1
Planificación de Sprint 0 ..... 2
  Tareas ..... 2
  Otros temas ..... 2
Resumen de Horas ..... 3
  Horas Comprometidas ..... 3
  Horas realmente trabajadas y en qué ..... 3
Retrospectiva ..... 4
  ¿Qué ha ido bien? ..... 4
  ¿Qué podría mejorarse? ..... 4
Revisión ..... 5
  Trabajo completado ..... 5
  Pendiente para otros sprints ..... 5
Gestión de tareas ..... 6
  Dinámica de trabajo ..... 6
  Selección de herramientas ..... 6
Repositorio de Código Fuente ..... 7
  Selección de la herramienta ..... 7
  Flujo de trabajo ..... 7
  Archivos de documentación ..... 7
Selección Base de Datos ..... 8
  Base de datos de documentos ..... 8
  RavenDB ..... 8
Generación de la documentación ..... 9

# Planificación de Sprint 0

## Tareas

- Definición de la arquitectura y diseños generales, estrategia técnica de la aplicación (3 hrs)
- Selección de la herramienta a utilizar para repositorio del código fuente (4 hrs)
  - Merge del branch de documentación (1 hrs - Andrés)
  - Discusión y pruebas (3 hrs) - terminado
- Selección de la herramienta a utilizar documentación (10 hrs)
  - Investigación de herramientas de conversión de markwdown a PDF / Investigación de herramientas de conversión de markdown a HTML / Investigación de herramientas de conversión de HTML a PDF / Investigación de configuración de dichas herramientas (6 hrs)
  - Decisión y capacitación sobre las herramientas (1 hr)
  - Configuración de dicha herramienta (3 hrs)
- Desglosar user stories (12:30 hrs)
  - Separar backlog en tres partes para generar user stories (JD - 30 mins)
  - Escribir user stories (6 horas - todos)
  - Reunión para asignar story points a los mismos (3 hrs - todos)
  - Reunión para asignar prioridades a los mismos (3 hrs - todos y Gabriel)
- Investigación de la herramienta a utilizar para gestión de tareas (7 hrs)
  - Investigación (6 hrs, todos)
  - Discusión (1 hr)
- Selección del sistema de base de datos a utilizar (7 hrs)
- Migrar Product Backlog al sistema de tracking del proyecto (1 hrs)
- Creación de los entornos de trabajo y su configuración – investigación de las tecnologías (~40 hrs, todos)

## Otros temas

- La idea que Gabriel se vaya involucrando cada vez más.
- Usaremos GitHub/Git para repositorio de código fuente y documentación.
- Matías tendrá problemas de conectividad, pero eso es algo que el repositorio puede manejar.
- Por lo general trabajaremos los fines de semana.
- Miércoles serán los días de reuniones.
- Martes, Jueves y Viernes cada uno debe enviar un email sobre su progreso, en qué va a trabajar y cómo va a proximar su trabajo. Se espera feedback de este tipo de mails. Los mails de cada uno deberían estar enviados durante la mañana, deberíamos tener todo el feedback resuelto para la noche. En el caso del viernes, hay plazo a cerrar los temas hasta el Sábado a la mañana.
- Trataremos que las reuniones de Planificación de Sprint duren 1:30 hs.
- Trataremos que las reuniones de Revisión de Sprint duren 1:30 hs. (Podría variar por el demo)
- Trataremos que las reuniones de Retrospectiva duren 30 mins.
- Esperamos no tener reuniones de auditoría no seguidas.
- Ante cada pull request al repositorio central, habrá dos revisores. El primero aceptará / rechazará el pull request. Si la persona lo acepta, el segundo revisor entonces lo cierra y hace el merge. Siempre dejaremos un comentario explicando qué se está haciendo y por qué.

# Resumen de Horas

## Horas Comprometidas

*Se incluyen las reuniones de planificación, revisión y retrospectiva*

- JD 30:00
- Matias 30:00
- Andres 30:00

## Horas realmente trabajadas y en qué

- JD (total 34:00)
  - Reuniones 11:00
  - Documentación (incluyendo mails) 4:00
  - Aprendizaje/investigación 19:00
- Matias (total 30:00)
  - Reuniones 7:00
  - Documentación (incluyendo mails): 1:30
  - Aprendizaje/investigación: 21:30
- Andres (total 30:30)
  - Reuniones 11:00
  - Documentación (incluyendo mails) 8:30
  - Aprendizaje/investigación 11:00

# Retrospectiva

## ¿Qué ha ido bien?

- La organización del grupo se va dando de forma natural y eficientemente.
- La comunicación es buena.
- Las reuniones de análisis parecen estar resultando bien.
- Fuimos bastante acertados en las horas comprometidas para trabajar

## ¿Qué podría mejorarse?

- Muchas tecnologías y prácticas nuevas nos hacen trabajar más lento de lo, tenemos que tener cuidado de no continuar agregando riesgos de este tipo.
- Los emails de los días Martes, Jueves y Sábados en reemplazo de los mails diarios de Scrum no terminan de ser una rutina y no llegan a ser totalmente útiles.
- Es difícil medir exactamente el tiempo invertido en cada tarea.
- Hubo una cierta dispersión, se realizaron tareas útiles fuera de las planeadas pero creemos que deberíamos atenernos más a lo planeado para tener resultados mas previsibles y cercanos a los deseos del cliente y del resto del grupo

# Revisión

## Trabajo completado

- Selección de la herramienta a utilizar para repositorio del código fuente.
  - Listo y documentado en el documento Detalles-Repositorio-de-Codigo-Fuente
- Desglosar user stories
  - Tarea realizada satisfactoriamente, aunque queda pendiente rankear individualmente cada item del *Product Backlog* y posicionar la *línea de fin de proyecto*.
- Investigación de la herramienta a utilizar para gestión de tareas
  - Listo y documentado en el documento Detalles-Gestion-de-Tareas
- Selección del sistema de base de datos a utilizar
  - Listo y documentado en el documento Detalles-Seleccion-Base-de-Datos
- Definición de la arquitectura y diseños generales, estrategia técnica de la aplicación
  - Si bién no se definió formalmente la arquitectura, los integrantes ya tenemos una idea más unificada de como debería ser y creemos que se irá definiendo naturalmente a medida que se implementen los user stories.
- Selección de la herramienta a utilizar documentación
  - Se encontró la herramienta adecuada y la forma de utilizarla, pero aún no se llegó a lograr la automatización esperada ni se documentó.

## Pendiente para otros sprints

- Rankear individualmente cada item del *Product Backlog*
- Posicionar la *línea de fin de proyecto* en el *Product Backlog*
- Documentar que herramientas se seleccionaron para generar la documentación para *entregas*.
- Documentar la forma en que se generarán las *entregas*.
- Lograr una automatización mayor de la generación de las *entregas*.

# Gestión de tareas

## Dinámica de trabajo

Para definir la dinámica de trabajo, nos basamos en la de uno de los proyectos de la empresa que está utilizando *Scrum*.

La idea es mantener el *Product Backlog* con los *User Stories*, *Bugs* y tareas técnicas en una herramienta de gestión de proyectos. El *Product Owner* y los integrantes del equipo se encargarán de actualizarlo, refinarlo y priorizarlo durante el transcurso del proyecto.

Al comienzo de la reunión de *Planificación del Sprint* se identificarán los *User Stories* y las tareas mejor rankeadas como potencialmente a ser realizadas en el *sprint*. Luego, de acuerdo con las necesidades del cliente, posibilidades de realización, relación costo/beneficio y madurez de los *user stories* se seleccionarán para ser realizadas en el sprint y se dividirán en subtareas cuyo ciclo de vida no va más allá del sprint. Su desarrollo se seguirá en una de pizarra con notas adhesivas (o una alternativa digital) y no se volcarán al sistema de gestión de proyectos.

## Selección de herramientas

Dado que nuestro *Product Owner* está familiarizado con JIRA y que la idea a mediano plazo es que él sea el responsable principal del *Product Backlog*, decidimos utilizar dicha herramienta para mantenerlo.

En cuanto al manejo de las tareas dentro del *sprint* decidimos utilizar Trello ya que es muy dinámico y nos permite simular una pizarra física con etiquetas adhesivas.

# Repositorio de Código Fuente

## Selección de la herramienta

Si bien dos de los integrantes del equipo están familiarizados con otros sistemas de control de versiones (Subversion y Team Foundation Server) creemos que el desarrollo de este proyecto es una buena oportunidad para conocer y aprender a utilizar un sistema de control de versiones distribuido como lo es Git. Esto nos permitirá trabajar más libremente y desconectados en nuestros repositorios particulares. Almacenaremos y compartiremos el código en GitHub, lo cual nos permitirá edición y code reviews online, de una forma muy cómoda y dinámica.

## Flujo de trabajo

Contaremos con un repositorio principal para el desarrollo donde mantendremos la rama principal *deployable* en GitHub y cada uno de los integrantes del equipo tendrá su propio *fork* en sus respectivas cuentas de GitHub: matias78, alphagit y andresmoschini.

Cada integrante del equipo utilizará Git y GitHub de la manera que le resulte más cómoda y funcional, pero para integrar cambios al repositorio principal deberá realizarlo mediante un pull request a través de GitHub, el cual será verificado por los demás.

## Archivos de documentación

Si bien los archivos de documentación se almacenarán en el mismo repositorio, no esperamos que se siga el mismo flujo de trabajo que con los archivos de proyecto, ya que muchas veces los cambios solo consisten en reflejar lo ya acordado en otras reuniones. Por lo tanto se podrán realizar modificaciones directamente en el repositorio principal.

# Selección Base de Datos

## Base de datos de documentos

Creemos que la utilización de una base de datos de documentos se adapta a las necesidades de negocio ya que la mayoría de nuestros objetos de negocio representarán estructuras complejas como *Postulantes* o *Curriculumns*.

De esta manera los datos almacenados serán mucho mas parecidos a nuestros objetos de negocio permitiéndonos ahorrar mucho trabajo en mapeo objeto/relacional.

## RavenDB

RavenDB es una base de datos de documentos programada en .NET con APIs .NET, Silverlight, Javascript y REST. A diferencia de otros sistemas de este tipo, RavenDB permite transacciones y su API para .NET es totalmente *.NET friendly*.

Ya que no tenemos experiencia previa *real* en el uso de esta base de datos y que la misma está en constante desarrollo, antes de decidirnos a utilizarla, realizamos una prueba de concepto tratando de replicar posibles escenarios que se presentarán en el sistema. Los problemas surgidos se pudieron resolver de una forma elegante, y cuando fue necesario la comunidad de RavenDB mediante la [lista de correo](lista de correo RavenDB) fue de gran ayuda y el autor principal respondió rápidamente en su GitHub.

Estamos convencidos de que la utilización de esta herramienta nos permitirá tener un código más limpio y que favorecerá las buenas prácticas y un buen diseño en nuestro sistema.

# Generación de la documentación

Optamos entre parte de nuestro proceso, incorporar algún método de generación automática de formatos de documentación, para poder fomentar la generación de documentación de forma espontánea, sin la preocupación de tener que reformatear la documentación existente.

Para esto se evaluaron varias opciones, entre las cuales, una de las mejores oportunidades fue la utilización del formato Markdown para la escritura de la documentación. La simpleza de este formato permite poder interactuar con los textos de una forma natural sin perder tiempo en el formateado de las mismas.

Por otro lado, las herramientas encontradas, específicamente MultiMarkDown to PDF (MMD2PDF) permitió la generación de documentos PDFs a partir de la estructura basada en los documentos originales.

Al momento de escritura, la documentación puede estilizarse con la utilización de estilos CSSs sobre archivos HTMLs intermedios generados desde el Markdown directamente. Tras esta conversión a CSS, la estructura jerárquica del HTML permite incorporar en la versión PDF una tabla de contenidos automatizada, basada en los niveles originales de títulos escritos en Markdown.

Aún está pendiente como tarea la estilización específica de estos CSSs y la personalización de la tabla de contenidos generada, tarea que requiere de cierto análisis debido a que esta capacidad ha sido una capacidad añadida a uno de los subsistemas de MMD2PDF (wkhthmltopdf) en una versión muy reciente, y por tanto se requiere su prueba y análisis de integración con el otro subsistema de MMD2PDF.

Cabe destacar que todas estas herramientas se encuentran en el dominio open source y en actual desarrollo activo, permitiendo la contribución de la comunidad, y del equipo mismo, si esto fuera necesario.