Proyecto curso

“FORMACIÓN INICIAL JAVA/DIGITAL”
[Creación de una agenda online]
[Lucatic]

18/03/2019 (Ed.16)

ÍNDICE DE CONTENIDOS

Si algún grupo no va a trabajar, DE FORMA CONTINUADA, con pruebas unitarias, pruebas de calidad, generación de documentación, usar javadoc y realizar seguimiento por LeanKit…
que no se moleste en entregar el proyecto porque lo tiene SUSPENSO

Planteamiento

Tu empresa realiza el proyecto de una agenda interna que simule un LDAP de personas… y te necesitan (aunque la verdad es que no tengo muy claro por qué te han elegido... pero bueno… hay gustos para todo. Hay gente que se le va la pinza y apuesta por un junior… pero vamos, que si me preguntan a mí se lo dejo bien clarito… pero no quiero malmeter…).

La BBDD a emplear ya está desarrollada por otro equipo de trabajo.

Vuestro equipo **SÓLO** debe desarrollar la funcionalidad de la aplicación.

Lo único que necesitas saber, de momento para planificar, es que esa BBDD tiene estas tablas (no necesitas ver la BBDD. NO la necesitas, lechón)

- Persona
- Direccion
- Telefono
- Provincia

NOTA: No seas ni pazguato, si pusilánime, ni ñoño, ni pudibundo, ni timorato, ni papanatas, ni lerdo, ni mentecato... porque que yo sepa tú tienes móvil y en tu móvil tienes una agenda... así que no me seas mangurrian y no me preguntes que funciones crear en la agenda.

Objetivos marcados

El **objetivo principal** de este proyecto se centra en tres aspectos:

- Conseguir que el grupo de alumnos sepa **utilizar** y mezclar la **teoría** y las **técnicas** vistas durante el curso relacionadas con Spring, JPA, MySQL, Angular, etc.
- Ayudar a que el alumno **desarrolle** y **potencie** las **competencias** marcadas o previstas para este curso:

Análisis y resolución de problemas	Cuenta con mi hacha	Autonomía
Trabajo en equipo	Comunicación	~~Thug Life~~
Flexibilidad	Proactividad	Motivación
Tolerancia a la frustración	Iniciativa	Interés a tope, lo voy a petar
	Soy tu BAE	Capacidad de aprendizaje
	Responsabilidad	

- Aprender y entender el funcionamiento de una **metodología de trabajo** como **SCRUM**

Trabajo a realizar a lo largo del proyecto

Cada grupo de alumnos/as debe **desarrollar** y **programar** las distintas partes marcadas en la "Descripción del Proyecto" y realizarlas de acuerdo a las prioridades marcadas más adelante.

El equipo debe **usar la metodología SCRUM** y crear un backlog de productos (revisable y ampliable durante el proyecto), un backlog de tareas por cada sprint y los cuatro tipos de reuniones (reuniones diarias, planificación de sprint, revisión y presentación de producto).

La pauta básica para todo el proyecto es la misma:

Ante la duda… simplifica

Descripción del proyecto (Tareas de Programación)

Este proyecto tendrá dos partes técnicas claramente diferenciadas:

a. **Spring MVC + HTML/JSP/EL/Thymeleaf**

   1. **Parte Pública** (FrontOffice): es la propia agenda online. Muestra el listado de contactos y permite el acceso a diversas funcionalidades:

      a. Listado de contactos con accesos para editar/eliminar

      b. Alta de nuevos contactos

      c. Búsqueda de contactos (básica)

      d. Ficha de cada contacto. Vista detallada

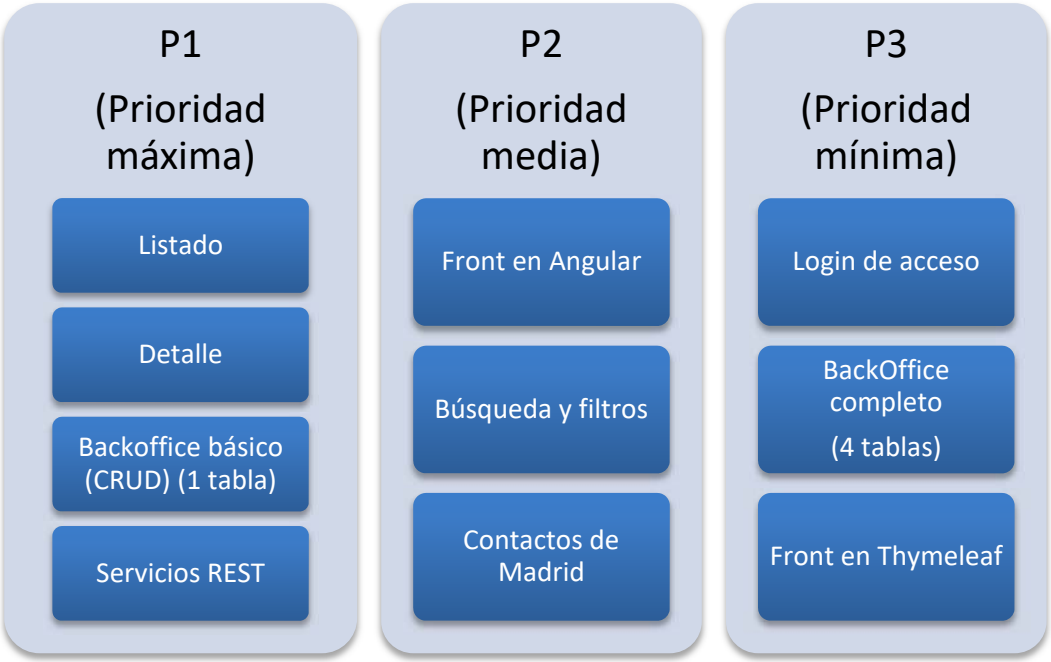
      Para esta parte no es necesario, en primera instancia, emplear login de acceso.

b. Spring REST + Angular

   1. Servicio Rest: Se crearán una serie de servicios REST que se puedan consumir desde un Front (Ej, borrar por ID, actualizar, guardar, recuperar el listado de contactos, etc.)

   2. Front Angular: Se creará el Front que permita consumir el servicio REST

Las tareas a realizar pueden dividirse en tres grupos dependiendo de su prioridad.



Prioridades del cliente y normas

Para este proyecto hay una serie de normas o prioridades. Cuantas más cumplas mejor:

- **Usar Spring** + **Hibernate** (JPA) + **SpringBoot**
- **Usar un front** con variantes para HTML.
  - Para Spring **MVC** usar HTML/EL/Thymeleaf, etc.
  - Para servicios **REST** usar **Angular**
- Si se realiza el sistema de **Login**, debe realizarse con **Spring Security**
- Emplear **logs de sistema** para mostrar información
- **Documentación Colaborativa**: Generar una documentación “formal/informal”
  - Posibles elementos: tareas realizadas, explicación breve de procesos y/o arquitectura, trabajo Sprint realizado, pantallazos de product backlog, etc.
- **Documentación JavaDoc**: Generada a partir de los scripts. Puede incluirse en la documentación colaborativa. Al menos una por integrante
- **Uso de repositorios**: para compartir las versiones de código.
  - Ideas: GitHub, GitLab, Bitbucket.
- Emplear una **herramienta digital** para la gestión del proyecto Scrum.
  - En este proyecto se ha seleccionado **LeanKit**

- **Pruebas unitarias:** define, de forma previa, varias pruebas unitarias.
  - Tienes que usarlas en todos los sprint. Pueden ser sencillas.
  - Proyecto erróneo si no se generan las pruebas antes de crear el método.
  - Puedes usar JUnit, DBUnit, Mockito, etc.
  - Para al menos 3-4 métodos/clases -> Usar **pruebas unitarias** para Spring
    - **TODOS** los integrantes deben saber cómo se hacen las pruebas unitarias
    - Posibilidad de hacer pruebas para **Angular**
- **Pruebas de calidad**: El código de un método/clase estará completo cuando pase estas pruebas. Si no se pasan esas pruebas el código no es completo

Para Clases

- Todos los paquetes van en minúsculas
- No se puede usar en el import .*
- En el import no pueden existir clases no utilizadas
- El código debe estar correctamente formateado
- Todo el código tiene que tener una cabecera en Javadoc que incluya: nombre de la clase, descripción, fecha, versión y autor.
- Toda clase (model) debe incluir setter/getter, constructor (default) y toString

Para métodos

- Las llaves de inicio del método están en la misma línea que la cabecera
- Todos los nombres de variables son descriptivos
- Usar sistema de logging para la información del programador

Tecnologías empleadas

Se pueden emplear todo tipo de configuradores, asistentes, wizards, template, etc. que faciliten el proyecto siempre y cuando el equipo responda de esas soluciones.

Cómo frameworks/librerías se recomiendan entre otros:

- **Spring** (y cualquier proyecto suyo)
- **JPA**/Hibernate
- **HTML5** y CSS
- **Angular**
- **Templates** Thymeleaf (posibilidad)

Entregas a realizar a lo largo del proyecto

a) Cada uno de los sprints marcados en la metodología de Sprint
b) El sprint final viene marcado por la entrega del proyecto.
c) La **presentación** se realizará junto al primer proyecto **el último día**
    a. Para la presentación cada equipo dispone de **media hora** en total (presentación y preguntas)
    b. El equipo vendrá **vestido de forma adecuada** para realizar la presentación
    c. Se recomienda que el equipo la haya **preparado de forma previa**
    d. Podrán utilizar cualquier recurso disponible: demos, **powerpoint** de apoyo, etc.

¿Cómo se valorará el proyecto?

Para valorar el proyecto se tendrán en cuenta tres aspectos

a. **Valoración técnica**: se revisará la calidad de la solución, elementos empleados, técnicas utilizadas, empleo de tecnologías, aplicación de los conocimientos vistos en clase

b. **Valoración competencial**: tomando como referencia algunas ideas basadas en competencias profesionales como el trabajo en equipo, análisis y resolución de problemas, flexibilidad, seguimiento de normas, etc.

c. **Directrices SCRUM**: haber seguido los parámetros indicados por la tecnología, seguir las necesidades del cliente, etc.

d. **Respecto a las directrices y normas planteadas**: respetar prioridades del cliente, directrices del profesor, etc.