

Proyecto curso

“FORMACIÓN INICIAL JAVA/DIGITAL”

[Creación de un sistema de citas]

[Lucatinder]

23/08/2019 (Ed.19)

ÍNDICE DE CONTENIDOS

Planteamiento.....	3
Objetivos marcados.....	3
Funcionamiento de Lucatinder	4
Sobre la BBDD	4
Trabajo a realizar a lo largo del proyecto.....	5
Descripción del proyecto (Tareas de Programación)	5
Prioridades del cliente y normas.....	6
Tecnologías empleadas	7
Entregas a realizar a lo largo del proyecto.....	8
¿Cómo se valorará el proyecto?.....	8

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



Eihhhhh... soy Hulio...

¿Quieres ser cómo yo y tener un amor en cada rincón? Con Lucatinder crearás (aunque sea falso) que eres como yo... ueeeaaahhhh!!.

Y es que los ~~leche~~ programadores de Lucatic no tienen tiempo para el amor y como son unos frikis no ligan ni a tiros y el profe ha decidido crear una aplicación para promover las citas entre alumnos y alumnas según sus gustos informáticos y te necesitan para programar (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... ueeeaaahhhh!!).

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, Rest, 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

Trabajo en
equipo

Flexibilidad

Tolerancia a la
frustración

Cuenta con mi
hacha y con
Hulio

Comunicación

Proactividad

Iniciativa

Soy tu BAE

Responsabilidad

Autonomía

~~Thug Life~~

Motivación



Interés a tope, lo
voy a petar

Capacidad de
aprendizaje

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

Como ves la actividad principal no es, en ningún caso, terminar toda la aplicación completa

Funcionamiento de Lucatinder

- 1) La primera pantalla que aparecerá será para darse de alta o bien para indicar el código de cliente (en el futuro se puede mejorar con un login más avanzado).
 - a. La persona se dará de alta y formará parte de la lista de PERFILES. El cliente siempre podrá modificar sus datos o darse de baja cuando quiera.
- 2) Para completar el perfil, al cliente se le pedirán datos (nombre, genero, edad, descripción y lista de gustos informáticos –opcional-) y gustos para realizar las futuras búsquedas (género y rango de edad)
- 3) Una vez que accede a su cuenta, al cliente le aparece de forma automática, una búsqueda de al menos 20 citas (como inicialmente no existen, se creará un proceso automático para crear perfiles falsos y registrarlos).
 - a. El cliente puede hacer clic (swipe a la izquierda) en  para descartar (y nunca aparecerán de nuevo) o clic (swipe a la derecha) en  para aceptarlas.
 - b. Si cada uno de los usuarios hace un “like” al otro, ocurre un “match” y ya se puede hablar con esa persona.

NOTAS (para el sistema de selección)

- Los perfiles que aparecen para elegir, incluyen una foto (opcional), el nombre, edad, población y una descripción de pocas palabras (opcional).
- **Algoritmo Básico:** se muestran 20 perfiles cualesquiera
- **Algoritmo Avanzado:** El algoritmo de selección da prioridad a estos 2 criterios:
 - Si un cliente elige un contacto, cuando ese perfil entre en su cuenta, el usuario 1 aparece entre los perfiles de su selección.
 - Contactos de la misma población.

Sobre la BBDD

La BBDD a crear dispone de 5 tablas principales (más las auxiliares necesarias)

- **PERFILES:** Guarda los datos de todos los usuarios registrados, incluyendo sus datos principales, código de cliente y sus datos para realizar las búsquedas.
- **MATERIAS:** Listado de materias informáticas. Una materia puede ser elegida por varios perfiles.
- **CONTACTOS:** Listado de Perfiles cuyo usuario 1 ha hecho “like” al usuario 2.
- **DESCARTES:** Listado de Perfiles cuyo usuario 1 ha descartado al usuario 2.
- **MATCH:** Listado de Perfiles cuyos usuarios se han hecho “like” mutuamente.

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 “Funcionamiento” 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 + Thymeleaf

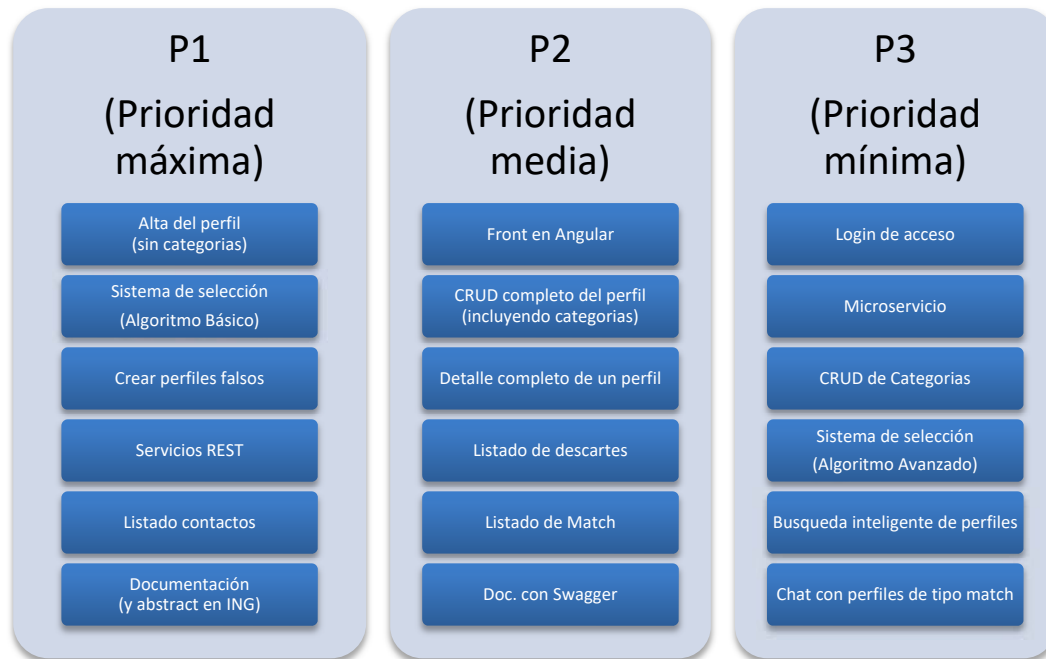
1. **Parte Pública** (FrontOffice.. no hay Back): es el propio Lucatinder. Accede al perfil del usuario y permite el acceso a diversas funcionalidades:
 - a. Mostrar listado de 20 perfiles al arrancar.
 - Si no existen debes crear perfiles automatizados falsos para mostrar (ver <https://github.com/DiUS/java-faker>)
 - b. Alta, modificación y detalles de mi perfil.
 - c. Sistema de selección  

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 y que repliquen la funcionalidad MVC.
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 + CSS + Thymeleaf
 - Para servicios **REST** usar **Angular**
- Si se realiza el sistema de **Login**, realizarlo con **Spring Security, jwt, Okta, Oauth2 ,...**
- 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.
 - Realizar un abstract de al menos 20 líneas en inglés
- **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 (GitHub, GitLab, Bitbucket)
- El **microservicio** está sólo de forma experimental y no hay que incluirlo en el backlog
- Emplear una **herramienta digital** para la gestión del proyecto Scrum: **Leankit**

- **Pruebas unitarias:** define, de forma previa, varias pruebas unitarias.
 - Tienes que usarlas en todos los sprint. Pueden ser sencillas.
 - Es erróneo si no se generan las pruebas antes de crear el método o funcionalidad (deberías pensarla y diseñarla de forma previa)
 - Puedes usar JUnit, DUnit, Mockito, etc.
 - Para al menos 3-4 métodos/clases -> Usar **pruebas unitarias** para Spring
 - **TODOS** los integrantes deben saber cómo hacer las pruebas unitarias
 - (Opcional) 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**

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 **el último día**
 - a. Se presentarán ambos proyectos: Proyecto 1 y Proyecto 2.
 - b. Para la presentación cada equipo dispone de **media hora** en total (presentación y preguntas)
 - c. El equipo vendrá **vestido de forma adecuada** para realizar la presentación
 - d. Se recomienda que el equipo la **prepare de forma previa**
 - e. 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.