TTPS opción Ruby

Trabajo integrador

Este documento define el alcance del trabajo integrador obligatorio con el cual los alumnos podrán obtener la cursada de la materia.

El contexto

Una startup platense te contactó para que implementes la idea que le da origen: implementar un servicio on-line de autogestión de listas de tareas. Para esto, quieren comenzar con un producto abierto basado en el modelo de servicios *freemium*: permitir utilizar abiertamente la aplicación web, y ofrecer funcionalidad avanzada para planes de pago.

Con el fin de comenzar a recibir feedback de usuarios reales lo antes posible, comenzarán con una idea rápida y sencilla de implementar que luego irá creciendo y evolucionando hacia el conjunto final deseado de servicios: el uso de la aplicación será sin autenticación, permitiendo a cualquier persona crear listas y gestionar tareas, sin necesidad de registrarse.

A continuación tenés el listado de requerimientos que se te solicitan para esta primera etapa del desarrollo, los cuales deben estar implementados en su totalidad para el Lunes 12 de diciembre de 2016.

Los requerimientos

Se te pide que desarrolles una aplicación web para gestionar listas de tareas.

Cada lista de tareas será públicamente accesible y modificable sin necesidad de autorización alguna, y tendrá los siguientes atributos:

- Un nombre descriptivo.
- Una URL única, basada en el slug del nombre que posea la lista.
- Un conjunto (inicialmente vacío) de tareas.
- Fechas de creación y última modificación.

El acceso a las listas será mediante la URL única que cada lista posea: la lista llamada `Cosas que dejé para mañana` tendrá como slug `cosas-que-deje-para-manana`, a partir del cual se formará una URL única (cuyo patrón queda a definir en tu implementación). Cualquier persona con esa URL podrá acceder a esa lista y gestionar las tareas que ésta posea.

Por su parte, las tareas pertenecen a una lista y podrán ser de diferentes tipos, a partir de los cuales se determina la información que se puede cargar para éstas. Los tipos posibles, y la información asociada a cada tipo, son:

- **Tareas simples:** son las clásicas tareas *TODO*, una tarea que posee dos estados posibles: pendiente o hecha. Sus atributos son:
  - Descripción: texto breve descriptivo de la tarea (máximo 255 caracteres).
  - Estado: Pendiente o hecha.
  - Prioridad: alta, media o baja.
- **Tareas temporales:** son tareas que sólo tienen razón de ser dentro de un periodo temporal. Sus atributos son:
  - Fechas de validez: un rango de fechas (inicio y fin) entre las cuales la tarea debe ser mostrada. Fuera de ese rango temporal la tarea no debe aparecer en la lista.
  - Descripción: texto breve descriptivo de la tarea (máximo 255 caracteres).
  - Estado: Pendiente, hecha o expirada. La tarea se marcará como expirada cuando se detecte (al mostrar la lista que la contiene) que la fecha de fin de validez ya ha pasado.
  - Prioridad: alta, media o baja.
- **Tareas largas:** son tareas similares a las simples, que pueden marcarse como “en curso” mientras se las está realizando, e indicar un porcentaje de avance. Sus atributos son:
  - Descripción: texto breve descriptivo de la tarea (máximo 255 caracteres).
  - Estado: Pendiente, en curso o hecha.
  - Prioridad: alta, media o baja.
  - Porcentaje de avance: un porcentaje (entre `0%` y `100%`) que indica el grado de avance sobre la tarea.

En todos los casos, la prioridad se utilizará para ordenar las tareas al mostrarlas, de más a menos prioritarias.

Los listados de tareas se presentarán mostrando toda la información posible, representada de la manera que consideres idónea para su mejor entendimiento.

Los clientes han indicado que quieren que la visual de la aplicación sea sencilla y salga de lo clásico: no quieren una aplicación con módulos CRUD (o “ABM”, como se los suele llamar en español), sino que quieren un diseño más ágil de utilizar y más amistoso. La implementación concreta de esto queda a tu criterio.

A fin de unificar criterios, los clientes han armado el siguiente *storyboard* detallando el flujo esperado y la funcionalidad a incluir en cada página de la aplicación:

- Pantalla inicial
  - Se puede crear una lista ingresando su nombre. Lleva a la *Vista de lista*.
  - Se deben mostrar, en caso de tener esta información, una lista con las últimas 5 listas creadas desde el mismo navegador desde el que está ingresando el usuario (utilizar la sesión del navegador), cada una con un vínculo para acceder a la lista de tareas (link a *Vista de lista*).

- Vista de lista
    - Se debe mostrar:
    - el nombre de la lista,
    - la fecha de creación de la lista,
    - la fecha de última modificación de la lista (se entiende por *modificación*: cualquier cambio realizado a algún dato de la lista y/o a los ítems que esta contiene),
    - un link a la lista (para copiar su dirección),
    - las tareas de la lista, con la mayor cantidad de información posible acorde al tipo de cada tarea, ordenadas por prioridad descendentemente.
    - Se debe permitir:
    - modificar el nombre de la lista (esta modificación no debe afectar la URL única de la lista),
    - agregar ítems a la lista,
    - modificar cualquiera de los ítems existentes,
    - volver a la *Pantalla inicial*.

Las restricciones

La aplicación debe:

- Ser versionada utilizando Git.
- Estar implementada en el framework Ruby on Rails en su última versión estable (al momento de publicación de este trabajo es `5.0.0.1`).
- Persistir los datos en alguna base de datos relacional (SQLite, MySQL, *choose your poison*).
- Utilizar *Single Table Inheritance* (STI) para modelar la jerarquía de tipos de tareas.
- Tener tests de unidad, al menos, para las siguientes situaciones:
    - La creación de una nueva lista:
    - Sin nombre.
    - Con nombre que respete la unicidad de los slugs.
    - Con nombre que cause un conflicto de unicidad de los slugs.
    - La creación de una nueva tarea:
    - Sin ningún dato.
    - Con datos válidos.
    - El ordenamiento de tareas:
    - De diferentes tipos, con diferentes prioridades.
    - La creación de una tarea temporal con un rango de validez invertido (fecha de inicio mayor a fecha de fin).
    - El pasaje de una tarea temporal a estado “expirada”.
    - La actualización de una tarea larga con porcentajes dentro y fuera del rango válido.
- Tener documentación de cómo preparar el ambiente para su correcto funcionamiento.
- Tener documentación de cómo ejecutarla.
- Tener un set de datos inicial (`seeds`) que contenga cualquier dato básico para su funcionamiento y una lista llamada `Lista 0` que contenga al menos 2 tareas de cada tipo.

La entrega debe:

- Realizarse de manera individual.
- Hacerse enviando el link al repositorio del proyecto en GitHub, BitBucket, GitLab o el hosting de Git que desees utilizar, y el link a la aplicación corriendo en Heroku (este último, según corresponda[1]).
- Contener toda la información y documentación necesaria para ponerla en funcionamiento en un archivo `README.md`.
- Cumplir, como mínimo, con todas las necesidades detalladas en la sección **Los requerimientos**.
- Definir y cumplir exitosamente con todos los tests de unidad pedidos.

> **Nota 1:** tené en cuenta que si vas a utilizar un repositorio privado, los integrantes de la cátedra necesitaremos acceso al mismo, para lo cual necesitás escribir a `ncuesta@info.unlp.edu.ar` indicando el hosting que vas a utilizar, para coordinar los datos de los integrantes que debés autorizar.
>
> **Nota 2:** el despliegue de la aplicación en Heroku será opcional en la primera fecha de entrega (con límite el 12/12/2016), y será obligatorio en la segunda fecha de entrega (con límite el 30/01/2017).

Algunas consideraciones adicionales:

- No se calificará la estética de la aplicación web, pero sí se espera un mínimo de criterio para mejorar la usabilidad de la misma.
- Queda a tu criterio interpretar qué atributos son requeridos en cada caso, reflejando esto en validaciones del modelo de datos, valores por defecto y restricciones de la base de datos.