Trabajo Práctico Especial

Programación Orientada a Objetos

1er Cuatrimestre 2015

Nombre del Proyecto: TrekApp

Fecha de Entrega: 18/06/2015

Integrantes:

- Natalia Navas
- Daniel Kochian
- Ignacio Cifuentes
- Matías Buscaglia

Objetivos y resumen:

El proyecto consiste en hacer una aplicación destinada a mochileros para fomentar el encuentro entre personas y la experiencia que esto representa.

El proyecto debe cumplir:

1. Tener un sistema de usuarios con un perfil con datos sobre la persona. Entre los datos se deben tener, aparte de información personal, un rating asignado por sus compañeros de viajes pasados.
2. Manejo de grupos por parte del administrador de estos. Este administrador podrá aceptar o rechazar solicitudes, decidir cuántos cupos tiene el grupo y agregar filtros para la gente que se quiera unir al grupo.
3. Uso de la api de Google Maps. Se busca mostrar la locación de los usuarios dentro de un grupo en tiempo real como también dar la posibilidad de un filtro que represente la distancia entre un usuario y el administrador del grupo.
4. Sistema de calificación por parte del administrador de un grupo luego de finalizar un viaje. Se debe poder calificar varios aspectos y asignar un rating que luego se deberá ver reflejado en el perfil de cada usuario calificado.
5. Tener una estructura Model-View-Controller para lograr un buen desacoplamiento entre la parte de interfaz de usuario y el modelo.
6. Simular un servidor utilizando herramientas tales como sqlite para poder almacenar los datos.

Cumplimiento de los objetivos:

Cumplimiento de los objetivos y decisiones al intentar cumplirlos (los números representan los mismos objetivos de arriba):

1. Se implementó completamente.
2. Se implementó completamente.
3. Debido al tiempo y la complejidad de implementación se decidió no utilizar la api de Google Maps y que cada usuario tenga guardada sus coordenadas, las cuales se actualizan cuando uno inicia sesión.
4. Se implementó completamente.
5. Creemos que se logró implementar completamente (ver Página 3).
6. Debido a problemas con la implementación utilizando sqlite se decidió que se almacena todo en memoria.

Estructuración del proyecto y decisiones tomadas:

La estructuración del proyecto fue la parte más difícil de hacer dadas las siguientes cuestiones:

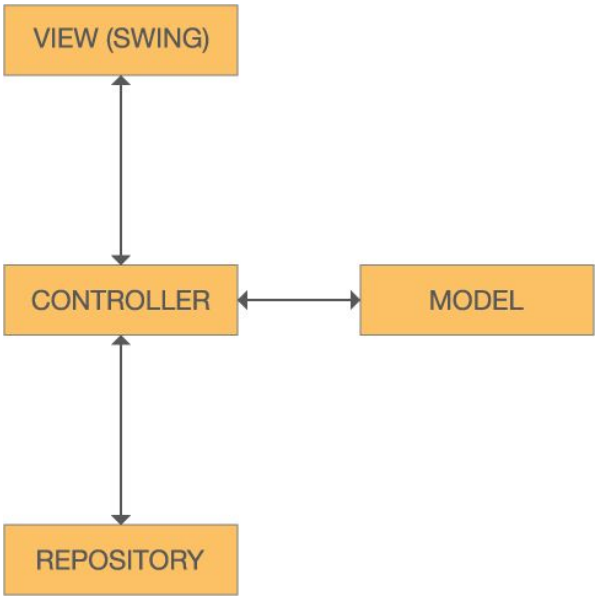
- El desacoplamiento utilizando el modelo Model-View-Controller (ver Página 4).
- El lograr restringir el acceso a métodos según los derechos administrativos del usuario loggeado. Para lograr esto se implementaron dos tipos de controladores para cada parte del modelo con la que se quiere interactuar. Por ejemplo, existe un controlador llamado CurrentProfileController que hereda de otro controlador llamado ProfileController, el ProfileController contiene solo métodos para obtener información del modelo (un profile) y el CurrentProfileController contiene, aparte de los métodos del ProfileController (por herencia), métodos para modificar el modelo (un profile). Esto permite que al iniciar sesión el usuario loggeado solo pueda modificar su perfil y ver información del de los demás. Se hizo la misma implementación con Group y Trip.
- El lograr que los controladores operen con el modelo sin implementar lógica que debería aparecer en el modelo. Para esto se pensó de la siguiente forma: La lógica que depende de cuál sea la implementación de los derechos administrativos para modificar y/o realizar acciones sobre el modelo la manejan los controladores ya que estos tienen el conocimiento de cuál es el usuario loggeado y sus derechos sobre cómo modificar el modelo. Luego toda la lógica que es propia del modelo y que no cambia según la implementación de los derechos administrativos se implementó en el modelo.
- El lograr qué haya solo un usuario loggeado y validar que haya una sesión en marcha. Para lograr esto se implementaron dos clases singleton, Application y Session. Session se ocupa de validar que existe una sesión y de guardar quién es el usuario loggeado. Application se ocupa de generar controladores para que use la View en el modelo Model-View-Controller. Cada vez que se quiere hacer una modificación en el modelo se valida que haya una sesión activa.
- El lograr que al modificar el modelo, también se modifiqué la base de datos. Para lograr esto cada vez que se modifica algo del modelo los

controladores llaman a un método que actualiza en el repositorio el objeto modificado en el modelo.

- Se decidió implementar el guardado en la memoria de la aplicación dado que tuvimos problemas para la completa implementación del servidor en sql (no se logró guardar Groups).

Implementación del Model-View-Controller:

A continuación se presenta un diagrama aproximado de cómo se desacoplan las partes del proyecto:



Model:

Aquí se encuentran objetos con su lógica que luego van a estar representados en las vistas de la aplicación. Se destacan el Profile (con todos los datos de un usuario), el Group (contiene miembros, un Trip, solicitudes para ingresar al grupo, etc) y Trip (contiene datos del viaje).

Controller:

Contiene los controladores con los cuales van a trabajar las vistas. También contiene al Session y al Application que también van a ser usados por las vistas. Se destacan los dos ProfileControllers (interactúan con el Profile del modelo), los dos GroupControllers (interactúan con el Grupo del modelo) y los dos TripControllers (interactúan con el Trip del modelo). Cada controlador tiene una referencia directa a

una parte del modelo y esta referencia es inaccesible para las vistas. La Application contiene métodos para registrar nuevos Group, Profile y Trip, y para crear nuevos controladores. La vista al Application sólo le puede pedir un controlador del usuario loggeado, los demás métodos son internos para los controladores y/o Session y para la misma Application.

Repository: Se hizo una implementación de guardado en memoria de la aplicación.

View:

Contiene las vistas que muestran partes del modelo. Se implementó en Java Swing debido al tipo de proyecto, al no necesitar una interfaz más compleja. Algunas de las características principales son:

- Se internacionalizó el proyecto de manera tal que se puedan agregar en un futuro más idiomas. Por el momento se encuentran el español y el inglés. La internacionalización se hizo mediante la herramienta de Java ResourceBundle, la cual contiene un locale que define el idioma con el cual el usuario desea ver la interfaz gráfica.
- Se utilizó el DatePicker para desplegar un calendario en la interfaz, permitiendo al usuario elegir un día. Esta librería no fue creada por nosotros sino que fue modificada a partir de la que se encuentra en internet.

Cómo usar la aplicación (para implementar una nueva interfaz gráfica):

Opción 1:

1. Si ya se tiene un usuario se hace Session.logIn(nombreUsuario, contraseña) y si devuelve true es que se inició sesión con éxito.
2. Luego se puede hacer Application.getCurrentProfileController() y obtener el controlador del usuario loggeado.
3. Finalmente se pueden crear grupos o Trips con Application y con los Controllers que devuelven utilizar otros métodos.

4. Para hacer búsquedas de Trips y de Profile se utiliza el CollectionAndSearchController.
5.

Opción 2:

1. Si no se tiene un usuario se hace Application.registerUser(datos) y devuelve el CurrentProfileController del usuario recien creado.
2. Finalmente se pueden crear grupos o Trips con Application y con los Controllers que devuelven utilizar otros métodos.
3. Para hacer búsquedas de Trips y de Profile se utiliza el CollectionAndSearchController.