*Instituto Tecnológico de Costa Rica*

*Unidad de Computación*

Lenguajes de Programación

I Semestre 2016

*Proyecto de Java*

*Yorbi Gerardo Méndez Soto*

*Sede San Carlos*

*4 de Jun. de 16*

Tabla de contenido

[Introducción 3](#_Toc452842217)

[Análisis del problema 3](#_Toc452842218)

[Solución del problema 4](#_Toc452842219)

[Análisis de resultados 6](#_Toc452842220)

[Conclusiones 6](#_Toc452842221)

Tabla de illustraciones

[Figure 1: Interface Repositorio 4](#_Toc452842227)

[Figure 2: Question interface 4](#_Toc452842228)

[Figure 3: Uso de Bundle 5](file:///B:\Documents\I_Semestre_2016\Lenguajes%20de%20Programación\Proyecto%204\Docu_Java_Project.docx#_Toc452842229)

[Figure 4: Enlazamiento de variables de una clase con los widgets de la vista. 5](file:///B:\Documents\I_Semestre_2016\Lenguajes%20de%20Programación\Proyecto%204\Docu_Java_Project.docx#_Toc452842230)

[Figure 5: Diagrama de base de datos. 5](file:///B:\Documents\I_Semestre_2016\Lenguajes%20de%20Programación\Proyecto%204\Docu_Java_Project.docx#_Toc452842231)

# Introducción

Analizar los diferentes paradigmas de programación y escoger cual utilizar para enfrentar un problema ha sido una de las cosas que un Ingeniero en Computación debe de realizar previo a iniciar a programar en cualquiera de los paradigmas. Se tiene claro que los paradigmas sirven para enfrentarse a diferentes tipos de problemas, de hecho, se podría hasta realizar una solución para un problema en varios paradigmas sin embargo lo ideal es que se escoja el mejor paradigma para la resolución del problema que se plantea. El siguiente proyecto consiste en explotar aquellos beneficios que ofrece el paradigma orientado a objetos, los cuales son el uso de herencia, la genericidad, la abstracción, polimorfismo y encapsulamiento y buscar la solución de un problema de creación y evaluación de exámenes, realizando una aplicación móvil que se base en este paradigma

# Análisis del problema

Se debe de implementar un sistema que permita al usuario crear, modificar y evaluar exámenes. Los exámenes deben de ser almacenados en la base de datos interna SQLite que ofrece los dispositivos y deben de ser cargados de la misma. Los mismos exámenes deben de disponer de secciones, de los cuales deben de contener un conjunto de preguntas, de diferentes tipos, sin importar el tipo de pregunta. El sistema debe de permitir al usuario crear más de un examen, al menos una sección por examen y al menos un tipo de pregunta por sección. Los tipos de preguntas debe de contener una clase para cada una, con sus propiedades bien definidos. Se debe de utilizar paquetes para realizar la lógica de la aplicación de la interfaz de la misma.

Todas las preguntas deben de estar asociadas a una sección, de tal forma que se pueda recorrer las listas de manera simple invocando aquel comportamiento que sea común entre las clases, para este se debe de utilizar un Interface, así se podrá definir los métodos que deben de contener cada una de las preguntas o secciones que hereden de la clase interface.

Para evaluar un examen, el sistema de recorrer todas las secciones en orden de apariencia y se debe de evaluar todas las preguntas de cada sección, luego, al finalizar el examen, el programa debe de brindarle al usuario las estadísticas del examen, es decir, la cantidad de respuestas correctas, la cantidad de respuestas incorrectas, el porcentaje obtenido y la nota que obtuvo el usuario en el examen.

# Solución del problema

Para manejar las diferentes preguntas, se creó un interface llamado Question que identifica todos los métodos que deben de tener las clases que sean de tipo pregunta, ya que estas clases van a implementar el interface Question. El interface Question consta de 3 metodos, uno llamado getQuestions() que no que hace es obtener la pregunta del objeto, otro llamado evaluateQuestion() que evalúa la pregunta luego de que el usuario haya seleccionado una respuesta y el ultimo método llamado showQuestion() que consiste en generar un Fragment para la visualización de la pregunta y retornar ese fragment para su muestreo en la aplicación. En la figura 2 se puede apreciar la estructura del interface Question utilizado para las preguntas.

Para llegar a la solución del problema, se dio uso a un NavigationDrawer que consiste en la utilización de fragments durante la app en vez de activities para brindarle un entorno más amigable al usuario. El Navigation Drawer tiene un uso similar a lo que se muestra en la aplicación de Gmail, que consta de un menú lateral para las opciones que puede realizar el usuario.

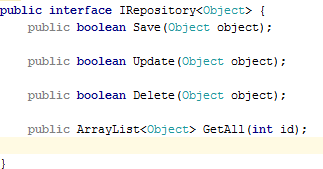
Ya que Android es de tipo MVC, Modelo Vista Controlador, para cada una de las vistas realizadas, se realizó una clase Java que maneje los datos de la vista y de la misma clase se accedió a los datos de la base de datos, por medio de un repositorio de datos (dependiendo del tipo de pregunta) de tal manera que el repositorio maneja la agregación, actualización y eliminación de los objetos. Estos repositorios deben de implementar una interface repositorio que define el esquema que deben de llevar las clases que la implementan. 

Figure 1: Interface Repositorio

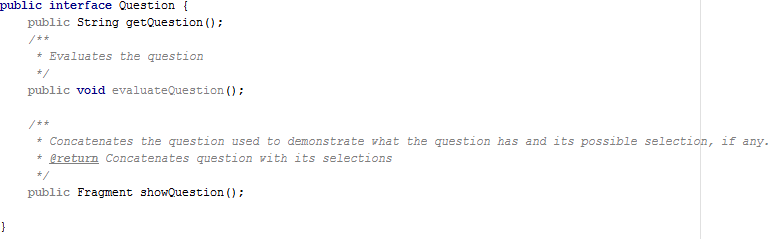


Figure 2: Question interface

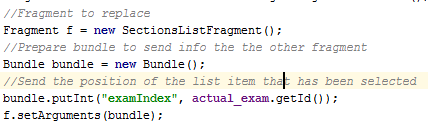
Para el traspaso de mensajes entre las clases se utilizó Bundle, ya que es la forma de realizar traspaso de mensajes en Android y con el mismo se puede de colocar la cantidad de parámetros que sean necesarios para pasar a otra vista o en este caso, a otro fragment que maneja una vista. Un ejemplo del uso de Bundle se puede apreciar en figura 3.

Figure 3: Uso de Bundle

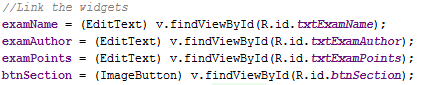
Para poder programar en Android es importante tener claro cómo funciona el modelo vista controlador. Nuestras clases en Java pueden ser nuestro controlador o bien pueden ser nuestros modelos. El controlador por lo general extiende de una actividad o de un fragment ya que va ser el encargado de manejar la comunicación entre la vista y el modelo. Ahora bien, todos los widgets que se desean manejar dentro del controlador o que sean necesarios manejar para el controlador deben de ser creados dentro del controlador y enlazados para el manejo de acciones u obtención de texto de un widget, dependiendo del widget que se utilizó. En la figura 4 se aprecia mejor como se lleva a cabo este enlazamiento. V en esa imagen se refiere a la vista (el layout) que vamos utilizar.

Figure 4: Enlazamiento de variables de una clase con los widgets de la vista.

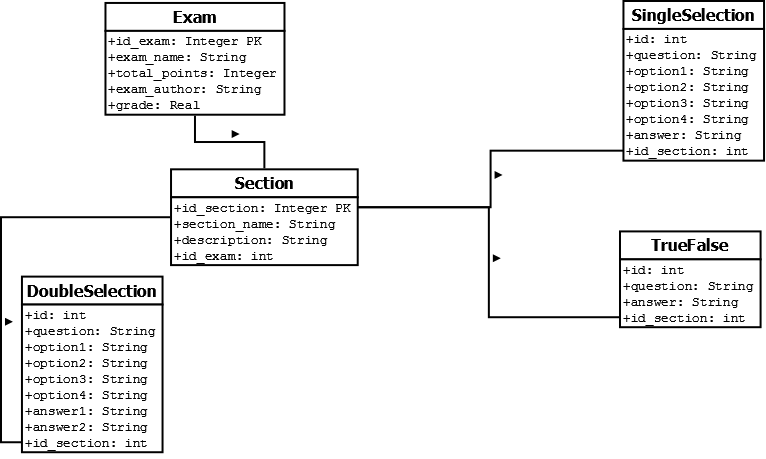


Figure 5: Diagrama de base de datos.

# Análisis de resultados

|  |  |  |
| --- | --- | --- |
| *Tarea/Requerimiento* | *Estado* | *Observaciones* |
| *Creacion de exámenes* | *Completo* | - |
| *Modificar exámenes – Agregar preguntas y secciones* | *Completo* | - |
| *Uso de base de datos interna* | *Completo* |  |
| *Evaluación de examen* | *Incompleto* | No muestra las preguntas |

# Conclusiones

El paradigma Orientado a Objetos es uno de los paradigmas más fáciles de entender, en mi caso, ya que es uno de los más utilizados en la actualidad y el mismo puede resolver un gran rango de problemas. Aunque no se pudiera finalizar por completo la parte de evaluación de los exámenes, la curva de aprendizaje de dicho lenguaje fue alta.

El desarrollo del proyecto brindo bastante conocimiento acerca de los Programas Orientado a Objetos y los beneficios que trae el mismo.