# Diseño de pruebas automáticas de losEstudiantes y AnthennaPod

Se va a diseñar e implementar una estrategia de pruebas sobre el sistema losEstudiantes y su aplicación web y AnthennaPod y su aplicación móvil para Android.

## Descripción de las aplicaciones

### losEstudiantes

Los estudiantes es una aplicación web en la cual los estudiantes de la Universidad de los Andes y la Universidad Nacional pueden calificar a sus profesores y así los departamentos y facultades tengan retroalimentación valiosa sobre cómo sus investigadores dan cátedra.

#### Detalles técnicos

La aplicación web está escrita en ReactJS, y el backend al parecer en NodeJS, que expone un servidor REST en express, al cual el front accede metiante el protocolo HTTPS.

#### Funcionalidades

Las funcionalidades de la aplicación son las registro y autenticación. búsqueda de un profesor y calificación de un profesor.

#### Casos de uso

- Registrarse en la aplicación.
- Ingresar a la aplicación.
- Buscar a un profesor por su nombre.
- Buscar a un profesor por materias que él dicta.
- Calificar un profesor.

### AntennaPod

AntennaPod es una aplicación open source Android que permite administrar podcasts. Los usuarios se subscriben a sus podcasts preferidos, y los pueden escuchar sus episodios via streaming o pueden descargarlos para escucharlos posteriormente.

#### Detalles técnicos

La aplicación web está escrita en Java, para la reproducción de audio a distintas velocidades utiliza Presto Sound Library. Los podcasts los consume de iTunes, FFYD, GPODDER o metiante una URL especificada por el usuario.

#### Casos de uso

- Suscribirme a un podcast.
- Escuchar un episodio vía streaming de un podcast.
- Descargar un episodio de un podcast.
- Escuchar un episodio descargado de un podcast.

# Arquitectura de pruebas

Debido al poco tiempo y recursos que se tiene, las pruebas se enfocarán en el core del negocio correspondiente a cada aplicación, se priorizarán los aspectos funcionales sobre los no funcionales.

## Recursos disponibles

Se cuenta con diez (10) horas máquina y diez (10) horas desarrollado.

## Tipos de pruebas

### End to end testing

Se decidió diseñar pruebas end to end para correr pruebas de sistemas y aceptación que validen que las funcionalidades de los sistemas bajo pruebas corran como fueron pensados.

Se ejecutarán pruebas end to end para cada caso enunciado de cada aplicación. Las pruebas se correrán en modo Headless para buscar minimizar el consumo de recursos computacionales y debido a que no es necesario hacer validaciones de interfaz.

Para la aplicación web se utilizará Cypress pues su simpleza se destaca para un proyecto tan remoto como lo es este. Una de las ventajas que tiene Cypress es que no se necesita que el desarrollador se preocupe por automatizar navegadores, como por ejemplo Selenium, pues Cypress tiene incluido uno propio.

Por otro lado, para las pruebas end to end de la aplicación móvil se utilizará Magneto, que es una de automatización open source para aplicaciones móviles Android.

### Random testing

Aunque suene muy tentativo diseñar pruebas de monkey testing manualmente, es decir, generando números aleatorios y eventos aleatorios para ejecutarlos sobre la interfaz gráfica, el tiempo no lo permite; lastimosamente, lo mismo ocurre con los Reapers. Se diseñará entonces pruebas de monkey testing con Frameworks que ya expongan funciones como la de llenar formularios y hacer clicks.

Para la aplicación web se utilizará gemlins.js debido a su facilidad y usabilidad, mientras que para la aplicación móvil se ejecutará la herramienta misma expuesta por Android Shell de pruebas monkey.

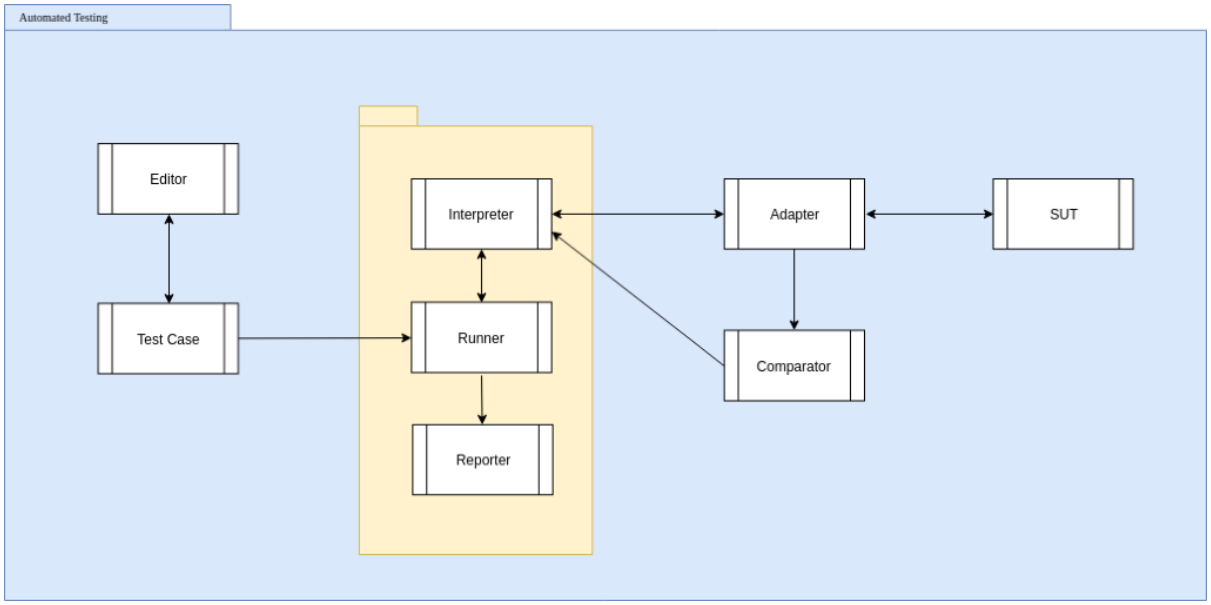
### Behavior Driven Development

Suponiendo que se tiene acceso directo con los clientes, se va a implementar Behavior Driven Development, esto para que estos tomen unn rol más activo en el proceso de pruebas y evitar que haya discordancias entre lo que espere el cliente y lo que haga el equipo de desarrollo.

Las herramientas que se van a utilizar para la aplicación web y la aplicación móvil son, respectivamente, Cucumber y Calabash.

## Vista general

La vista general de la arquitectura es la siguiente:




Para el caso de las pruebas end to end, los casos de prueba serán generados manualmente, así como los oráculos, mientras que los intérpretes, ambientes de ejecución y generador de reportes los aportará el framework de pruebas.

Para las pruebas monkey de la aplicación web serán con casos de uso manualmente dirigidos, mientras que para la aplicación móvil se dejará el monkey aleatoreamente explorar la aplicación; los oráculos de estas pruebas son simplemente el hecho que la aplicación crahsee o no.

Finalmente, para las pruebas BDT, el desarrollador genera los casos de pruebas y los oráculos manualmente, por su lado, el framework de pruebas provee el ambiente de ejecución, generador de reportes e intérpretes.