Práctica final

Crear un nuevo proyecto empleando la plantilla "Tabbed Activity". Los demás datos se pueden quedar por defecto.

Añadir la configuración de AnodridAnnotations, para lo cual habrá que modificar el `build.gradle` y el `AndroidManifest.xml`. Copiar los recursos que se dan en los respectivas carpetas de recursos.

De la `MainActivity` eliminar todo el contenido de la clase (ya que lo haremos manual) y poner la anotación `@EActivity(R.layout.activity_main)`.

En su respectiva vista (activity_main.xml) añadir en la Toolbar dos botones para cambiar el idioma añadiendo la siguiente configuración:

```
style="@style/Widget.AppCompat.Button.Small"
    android:layout_gravity="end"
    android:layout_weight="1"
```

El texto de cada botón se recuperará de un recurso y se deberá adaptar al idioma seleccionado (por ejemplo "ES" y "IN" para español y ejemplo "SP" y "EN" para inglés).

Estos botones serán instanciados y creado su setOnClickListener en el método `@AfterViews`. Cada vez que uno de estos sea pulsado se deberá actualizar el lenguaje de toda la aplicación, por el momento sólo el lenguaje de la toolbar. Para el cambio se empleará una animación se que seleccionará dependiendo de un `@BooleanRes` (para poder selecionar una animación u otra).

```
        bEnglishChange.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                changeToEnglish();
            }
        });
        bSpannishChange.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                changeToSpanish();
            }
        });

    private void changeToEnglish() {
        changeTextFromToolbar(english.getAppName(), english.getBSelectEn
    }

    private void changeToSpanish() {
        changeTextFromToolbar(spanish.getAppName(), spanish.getBSelectEn
    }
```

```
    private void changeTextFromToolbar(String app_name, String b_select_
        if (is_fade_in) {
            toolbar.startAnimation(fade_in);
        } else {
            toolbar.startAnimation(appear);
        }
        toolbar.setTitle(app_name);
        bEnglishChange.setText(b_select_en);
        bSpannishChange.setText(b_select_sp);
    }
```

A continuación vamos a crear el TextView personalizado. Para ello crear el paquete "views" y añadir la nueva clase `CustomTextView`. Ésta heredará de `android.support.v7.widget.AppCompatTextView` e irá anotada con `@EView`. Crear el constructor con los parámetros de entrada `Context context, AttributeSet attrs` y además añadir el siguiente método:

```
    public void setText(String hero, String actor, String desc, String l
        String infoToShow;

        if (EN.equals(language)) {
            infoToShow = "Avenger: " + hero + "\n" +
                    "Actor: " + actor + "\n" +
                    "Description: " + desc;
        } else {
            infoToShow = "Vengador: " + hero + "\n" +
                    "Actor: " + actor + "\n" +
                    "Descripción: " + desc;
        }

        setText(infoToShow);
    }
```

Posteriormente pasaremos a definir el objeto personalizado, para ello renombrar la `fragment_main.xml` por `custom_avenger_view.xml`. Poner un ImageView que ocupe todo el ancho de la vista y que además tenga una descripción (por ejemplo: `android:contentDescription="@string/app_name"`). A continuación, debajo de la imagen, poner el CustomTextView creado anteriormente, recordar que se debe emplear la clase generada por AndroidAnnotations, es decir `CustomTextView_`. Este objeto personalizado deberá ocupar todo el ancho de la pantalla, pero le podremos dar unos margenes laterales y superior de "`5dp`". Finalmente crear `5` botones debajo del anterior objeto y alineados horizontalmente. Cada botón tendrá la funcionalidad de cambiar el fondo de objeto personalizado que estamos creando.

El siguiente paso es crear la clase "CustomView" que será la lógica Java que trabaje con el "custom_avenger_view.xml" creado anteriormente. Éste debera estar

anotado con `@EViewGroup(R.layout.custom_avenger_view)` y deberá heredar de `RelativeLayout`. En el método `@AfterViews` se deberá instanciar todos los objetos creados anteriormente y adicionalmente crear un método "setInfo" que permita:

1. Cambiar la imagen del vengador.
2. Actualizar la información mostrada del CustomTextView empleando el método infoToShow.
3. Cambiar el fondo de cada uno de los botones.
4. Añadir el evento a cada botón para poder cambiar el fondo del custom_avenger_view.

Toda esta información deberá viajar en el objeto `HeroInfo`, esta clase se debe crear en el nuevo paquete `recoverinfo.dto`. Y tendrá la siguiente información (obtenida por los respectivos getters) con su respectivo constructor (que inicializará toda la información):

```
private int id = 0;
private String language = null;
private int image = 0;
private String hero = null;
private String actor = null;
private String description = null;
private int[] colors = null;
```

Una nueva aplicacion es la que debe orquestar toda esta información con la `MainActivity`. Crear la clase `RequestInfoActivity` en el paquete creado anteriormente `recoverinfo`. En esta nueva clase se deberán instanciar los beans de idiomas, los cinco arrays con los colores de cada vengador y además añadir el siguiente método que será el encargador de seleccionar le vengador de cuerdo al idioma:

```
public HeroInfo callHero(int heroId, String language) {
    HeroInfo heroInfo = null;

    switch (heroId) {
        case IRON_MAN:
            if (EN.equals(language)) {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            } else {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            }
            break;
        case SPIDER_MAN:
            if (EN.equals(language)) {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            } else {
```

```
                heroInfo = new HeroInfo(heroId, language, R.drawable
            }
            break;
        case CAP_AMERICA:
            if (EN.equals(language)) {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            } else {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            }
            break;
        case VIUDA:
            if (EN.equals(language)) {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            } else {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            }
            break;
        case THOR:
            if (EN.equals(language)) {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            } else {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            }
            break;
        case HULK:
            if (EN.equals(language)) {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            } else {
                heroInfo = new HeroInfo(heroId, language, R.drawable
            }
            break;
    }

    return heroInfo;
}
```

Finalmente, se ha empleado una clase con constantes que estará en le paquete `recoverinfo.constants` y será:

```
public class Constants {

    public static final int IRON_MAN = 0;
    public static final int SPIDER_MAN = 1;
    public static final int CAP_AMERICA = 2;
    public static final int VIUDA = 3;
    public static final int THOR = 4;
    public static final int HULK = 5;

    public static final String EN = "EN";
```

```
        public static final String ES = "ES";
    }
```

Y los beans de idiomas que se encontrarán bajo el paquete "beans" siendo estos:

English -> Todos los los recursos de string en inglés. Spanish -> Todos los los recursos de string en español.

A continuación actualizar la `activity_main.xml` con el `CustomView_` y dos botones (uno a la izquierda y otro a la derecha) que nos permitan avanzar/ retroceder entre los vengadores. Para concluir instanciar y definir los listener de los botones:

```
        ib_next.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                nextAvenger();
            }
        });
        ib_back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                backAvenger();
            }
        });

    private void nextAvenger() {
        String language = heroe.getLanguage();
        counterAvenger = (counterAvenger+1)%6;
        heroe = receiverAction.callHero(counterAvenger, language);
        customView.setInfo(heroe);
    }

    private void backAvenger() {
        String language = heroe.getLanguage();
        counterAvenger = (counterAvenger-1)%6;
        if (counterAvenger == -1) {
            counterAvenger = 5;
        }
        heroe = receiverAction.callHero(counterAvenger, language);
        customView.setInfo(heroe);
    }
```

El último paso que quedaría antes de poder ejecutar la aplicación es acualizar el idioma, para ello ir a los métodos `changeToEnglish` y `changeToSpanish` y añadir la siguiente línea:

```
    private void changeToEnglish() {
        changeTextFromToolbar(english.getAppName(), english.getBSelectEn
```

```
    //New line
    changeCustomView(Constants.EN);
}
private void changeToSpanish() {
    changeTextFromToolbar(spanish.getAppName(), spanish.getBSelectEn

    //New line
    changeCustomView(Constants.ES);

}

private void changeCustomView(String language) {
    if (is_fade_in) {
        customView.startAnimation(fade_in);
        ib_next.startAnimation(fade_in);
        ib_back.startAnimation(fade_in);
    } else {
        customView.startAnimation(appear);
        ib_next.startAnimation(appear);
        ib_back.startAnimation(appear);
    }

    heroe = receiverAction.callHero(heroe.getId(), language);
    customView.setInfo(heroe);
}
```