

## Proyecto Integrado: Desarrollo de una aplicación para Android, IOS, Web, Escritorio.



**CICLO FORMATIVA DE GRADO  
SUPERIOR  
DESARROLLO DE APLICACIONES  
MULTIPLATAFORMA**

**Juan Maqueda Vargas**

**José Iván Jiménez Rodríguez**

**IES AL-Andalus**

Curso 2013/2014

## INDICE

<b>INDICE</b>	<b>1</b>
<b>1. INTRODUCCIÓN</b>	<b>3</b>
<b>2. ANALISIS DEL ENTORNO</b>	<b>4</b>
2.1. Descripción de las oportunidades de negocio	4
2.2. Obligaciones fiscales, laborales y de prevención de riesgos del proyecto	8
2.2.1. Prevención de riesgos	8
2.2.2. Obligaciones laborales y sociales	8
2.2.3. Obligaciones fiscales	9
2.3. Posibles ayudas o subvenciones para el desarrollo del proyecto	13
2.3.1. Subvenciones	13
2.3.2. Ayudas	17
<b>3. ANALISIS DEL SISTEMA ACTUAL</b>	<b>18</b>
<b>4. SOLUCION PROPUESTA</b>	<b>18</b>
4.1. Diferentes tecnologías existentes para el desarrollo y puesta a punto de la aplicación que se desarrollara.	18
4.1.1. Comparativa de los distintos sistemas operativos para dispositivos móviles	18
4.1.2. Gráficas de estadísticas	21
4.1.3. Introducción a libGDX	24
4.2. Estudio económico. Valoración de presupuestos	37
4.3. Elección y justificación de las tecnologías seleccionadas	37
4.4. Recursos humanos y materiales necesarios para desarrollar el proyecto	44
<b>5. PLANIFICACION TEMPORAL DEL DESARROLLO DEL PROYECTO</b>	<b>45</b>
5.1. Desarrollo del proyecto	45
5.2. Lista de modificaciones y ampliaciones futuras que se podrían aplicar a la aplicación proyectada.	46
5.3. Plan de mantenimiento de la aplicación	47
<b>6. ESTUDIO DE VIABILIDAD DEL PROYECTO</b>	<b>48</b>
<b>7. DOCUMENTACIÓN DEL DISEÑO E IMPLEMENTACIÓN DE LA SOLUCION ADOPTADA</b>	<b>50</b>
7.1. Prototipo de la aplicación. Diseño de las interfaces	50
7.2. Diseño lógico de la aplicación a desarrollar, diagramación UML o de Flujo de datos.	50
7.3. Descripción modular del software a desarrollar	50
7.4. Diseño de la/s bases de datos, con sus diagramas correspondientes.	54
7.5. Otras estructuras de datos que se utilizan en la aplicación.	55
7.6. Estudio de la seguridad de la aplicación	56
<b>8. CODIGO FUENTE DOCUMENTADO</b>	<b>57</b>
<b>9. MANUAL DE CONFIGURACIÓN Y FUNCIONAMIENTO DE LA APLICACION</b>	<b>87</b>

10.	MANUAL DE USUARIO	87
11.	PLAN DE FORMACION A LOS USUARIOS DE LA APLICACIÓN	87
11.1.	Contenidos a tratar	87
11.2.	Tiempo estimado de impartición de cada contenido	87
12.	BIBLIOGRAFÍA Y FUENTES DE INFORMACION	88

## 1. INTRODUCCIÓN

El proyecto desarrollado ha sido una aplicación multiplataforma (iOS, Android y Escritorio) llamada BabyEduca.

El desarrollo de la aplicación se ha basado utilizando libGDX, una librería orientada a juegos, que permite codificar el núcleo en Java y compilar en las distintas plataformas soportadas.

BabyEduca es una aplicación destinada a fomentar el buen comportamiento en los niños y adolescentes. Desarrollada por un equipo de expertos en educación y psicología se ha concebido para convertirse en una valiosa herramienta para usarla tanto en casa como en el aula docente. De este modo, a través de una interfaz sencilla y amigable, BabyEduca ayudará a padres y educadores a gestionar esos comportamientos que quieren mejorar en sus hijos y alumnos, en base al uso de las recompensas más eficaces para cada caso.

Se pueden seleccionar las tareas o comportamientos deseados de una lista predefinida o incluir otros nuevos. Lo mismo puede hacerse con las recompensas. Por supuesto, es esencial dedicar cada día cierto tiempo con su hijo utilizando BabyEduca a fin de comprobar si los objetivos se han logrado para las tareas previstas.

No hay que olvidar que el éxito debe ser siempre recompensado, así que juegue con su hijo a obtener nuevos puntos arrastrando una estrella desde la canasta hasta la cuadrícula del calendario. Es muy importante que los niños interactúen con la aplicación como si estuvieran jugando con ella, y sus logros sean reforzados con expresiones positivas como: ¡bien hecho! ¡Lo hiciste muy bien!, ¡te mereces una estrella!

Así conforme el comportamiento de los niños vaya mejorando, el número de puntos irá aumentando y los premios estarán disponibles.

## 2. ANALISIS DEL ENTORNO

### 2.1. Descripción de las oportunidades de negocio

***“La innovación es lo que diferencia al líder de aquellos que le siguen”*** (Steve Jobs).

Las aplicaciones para móviles, tras varios años en proceso de incubación, se encuentran por fin en plena expansión.

Las oportunidades de negocio son colosales, con un mercado potencial de 30 millones de clientes tan solo en España.

Una de las particularidades de este gran nicho es que sus usuarios siempre llevan consigo el móvil y lo utilizan en todo momento y por cualquier motivo, lo que les hace especialmente atractivos desde un punto de vista de negocio.

Unos datos curiosos:

- En España hay más de 52 millones de teléfonos móviles, más móviles que habitantes.
- El 55% son teléfonos inteligentes: Android, iPhone, Blackberry, Windows Phone ...
- España es líder de Europa en cuanto a inserción de smartphone en el mercado.
- El e-commerce móvil ha cosechado un crecimiento del 140% en el último año.
- Los usuarios de teléfonos inteligentes compran 4 veces más que un usuario de escritorio.



## Desarrollo de aplicaciones multiplataforma

A continuación comentamos cinco tipos de aplicaciones con los modelos más interesantes:

- **Fidelización de usuarios:**

Se trata de aplicaciones en las que la empresa es referente en un área temática o, por ejemplo, en un área geográfica ofreciendo al usuario información o herramientas de utilidad que brindan un valor añadido. Este tipo de aplicaciones ayudan a aumentar la exposición de la marca y a mejorar su imagen.

Entre los principales beneficios que ofrecen este tipo de aplicaciones encontramos la exposición y mejora de la marca.

Un ejemplo real en este tipo de aplicaciones es el desarrollado por Martín Martín, una cadena de tiendas aragonesa que en el año 2012 patrocinó el desarrollo de la aplicación de las Fiestas del Pilar que ofrecía a sus usuarios información de los eventos disponibles durante las fiestas, localización e información de interés del lugar y un pequeño juego para generar virilidad en las redes sociales. Asimismo, la aplicación disponía de un apartado donde la empresa mostraba ofertas puntuales, ubicación de sus tiendas y enlaces a sus perfiles sociales. Esta herramienta permitió a la empresa mejorar su branding, llegar a nuevos clientes potenciales, incrementar ventas a través de ofertas especiales y aumentar su visibilidad en redes sociales.

Otras muchas empresas apuestan por el desarrollo de aplicaciones para fidelizar a sus clientes, como Nike con su aplicación Nike+ o Cepsa con su avisador de radares.



- **Canal de venta:**

Como adelantábamos al comienzo del artículo, el comercio móvil crece a un ritmo muy superior al comercio digital tradicional, siendo los usuarios móviles 4 veces más proclives a la compra online.

Recurrir a este modelo es muy recomendable si un porcentaje significativo de tus clientes se encuentran dentro del segmento de usuarios de Smartphone y/o consumen tus productos o servicios vía móvil. Antes de invertir en el desarrollo de una app de este tipo conviene evaluar si un sitio web optimizado para dispositivos móviles es suficiente.

## Desarrollo de aplicaciones multiplataforma

Las aplicaciones de canal de venta aprovechan la rapidez y sencillez de las transacciones móviles para transformar la impulsividad de sus compradores a ventas.

Un ejemplo de apps de canal de venta es Privalia, que ofrece a sus usuarios la posibilidad de realizar compras a través de teléfonos móviles.



- **Ingresos por venta o publicidad:**

Otro tipo de aplicaciones interesantes son aquellas cuyo modelo está enfocado a obtener ingresos por ventas de la propia aplicación o mediante publicidad incrustada.

Suelen tratarse de aplicaciones que desarrollan ideas innovadoras, de utilidad en contextos empresariales o que consiguen un hueco dentro del mercado del ocio.

Ejemplos muy conocidos de este tipo de aplicaciones son Whatsapp o Angry Birds, esta última ingresó 100 millones de dólares solo en 2011 mediante ventas de la aplicación, publicidad y merchandising.



- **Híbridas:**

También pueden desarrollarse apps híbridas que engloben diferentes modelos de negocio dentro de una misma aplicación. Por ejemplo, una aplicación dirigida a fidelizar usuarios que además obtenga ingresos por publicidad.

Piensa en el caso de un hotel de una ciudad turística que quiere diferenciarse de su competencia a través de una aplicación de valor añadido para los turistas de su ciudad, quizá un audio guía para visitar monumentos emblemáticos del lugar. Esta aplicación podría utilizarse a su vez como una plataforma de difusión de ofertas especiales del hotel, pero también como canal de publicidad para otros establecimientos relacionados (restaurantes, agencias de ocio, etc.) a los que podría cobrarse una cuota por aparecer en la aplicación.

- **Gestión de procesos internos:**

Aunque estas aplicaciones no siguen un patrón orientado a la generación de ingresos directos, las hemos tenidos en cuenta porque son de utilidad para aumentar la productividad interna de una organización y pueden utilizarse para mejorar los procesos de ventas.

Con este tipo de apps puedes llevar tu negocio dentro del teléfono móvil pudiendo, entre otras muchas acciones, consultar pedidos, gestionar clientes, consultar catálogos y listas de precios, etc.

Podemos tomar como ejemplo la empresa Quesos Vega Sotuélamos, que en el año 2012 desarrolló una aplicación de catálogo de productos para dispositivos iPad de uso interno con la que poder mostrar de forma ágil su catálogo de productos, consultarlo sin conexión, actualizarlo o generar pedidos para ser gestionados desde su departamento de ventas, entre otras funciones. Dicha aplicación se ha convertido para el departamento comercial de Quesos Vega Sotuélamos en una herramienta de trabajo indispensable.

SugarCRM, Dropbox o Evernote son una mínima muestra de aplicaciones disponibles en plataformas móviles dirigidas a mejorar la productividad de los negocios.



## **2.2. Obligaciones fiscales, laborales y de prevención de riesgos del proyecto**

### **2.2.1. Prevención de riesgos**

Los principales riesgos a los que están expuestos los trabajadores que desarrollan su actividad en oficinas y despachos son:

- Caídas y golpes contra objetos.
- Posturas y movimientos adoptados.
- Manipulación manual de cargas.
- Fatiga visual.
- Confort acústico.
- Confort térmico.
- Calidad del aire interior.
- Radiaciones y campos electromagnéticos.
- Factores psicosociales.

A continuación se desarrollan de forma más detallada cada uno de estos riesgos, y se proponen las medidas preventivas adecuadas al mismo.

En el siguiente enlace encontraras la documentación de este apartado para una mayor claridad del mismo:

[Manual Prevencion de riesgos laborales.pdf](#)

### **2.2.2. Obligaciones laborales y sociales**

Las obligaciones sociales y laborales surgen como consecuencia de la relación laboral que se establece entre la empresa y los trabajadores que prestan sus servicios en la misma, y de la existencia de una normativa específica en materia laboral.

Las principales obligaciones sociales y laborales del empresario son:

Antes de empezar la actividad:

- Inscripción de la empresa en la Seguridad Social para aquellos empresarios que vayan a contratar trabajadores
- Darse de alta en el Régimen de Autónomos (Empresarios individuales, Comunidades de bienes y sociedades)
- Dar de alta los trabajadores por cuenta ajena en el Régimen General de la Seguridad Social
- Comunicación de apertura del centro de trabajo.

## Desarrollo de aplicaciones multiplataforma

Después de comenzar la actividad:

Cumplir con la normativa laboral, entre la que destaca el Estatuto de los Trabajadores, el Convenio Colectivo correspondiente, las reglamentaciones de régimen interno y las normas sobre Seguridad Social.

En particular, cumplir las siguientes formalidades:

- Dar de alta los nuevos trabajadores por cuenta ajena en el Régimen General de la Seguridad Social.
- Respetar los derechos del trabajador.
- Formalizar por escrito los contratos de trabajo cuando así lo exija una disposición legal.
- Entregar a la representación legal de los trabajadores una copia básica de todos los contratos que deban celebrarse por escrito, a excepción de los contratos de relación laboral especial de alta dirección.
- Informar por escrito a los trabajadores sobre los elementos esenciales del contrato de trabajo y las principales condiciones de ejecución de la prestación laboral.
- Informar por escrito de cualquier modificación de los elementos del contrato al trabajador que resulte afectado.
- Comunicación a los Servicios Públicos de Empleo.

### 2.2.3. Obligaciones fiscales

#### ¿Cómo se declaran en Hacienda la venta de aplicaciones en Google Play?

Muchos de los desarrolladores de aplicaciones que posteriormente, las venden en Google Play. El problema que tienen la inmensa mayoría de estos desarrolladores, es la declaración de estos ingresos que generan estas ventas y el caso es que muchos están perdidos y no saben cómo cumplir correctamente con Hacienda y con la Seguridad Social.

El mecanismo legal para declarar estos ingresos en España y todo lo que vamos a exponer es válido única y exclusivamente para programadores que tienen su residencia fiscal aquí y que han identificado España como su país dentro de Google Play para comercializar la aplicación, siempre que hayan creado una cuenta en Google Play como personas físicas y no como sociedades limitadas.

### **Los ingresos en Hacienda de aplicaciones móviles, actividad empresarial:**

Tengamos siempre en cuenta que desde el punto y hora que cobremos un solo euro por una sola venta, nuestras ventas tienen la consideración de actividad empresarial para Hacienda y la manera correcta de declarar estos ingresos es tramitando nuestra alta como empresarios en la Agencia Tributaria.

Esta alta se tramita cumplimentando el modelo 037, declaración censal de alta de actividades económicas. Para todos los que no estén relacionados con cuestiones fiscales, lo mejor que pueden hacer para cumplimentar correctamente este modelo es dirigirse a su oficina de la Agencia Tributaria más cercana y que se lo cumplimenten o bien, dirigirse a una asesoría.

Una vez hemos tramitado el alta, trimestralmente tendremos que presentar (y pagar) en la Agencia Tributaria un modelo de pagos a cuenta en el IRPF, que es el modelo 130 y otro modelo que es la declaración trimestral de IVA, modelo 303. Para los pagos a cuenta, tendremos que calcular el total de ingresos (excluido el IVA) menos los gastos deducibles (que en nuestro caso no existen gastos deducibles) e ingresar trimestralmente el 20% del beneficio neto a Hacienda.

Esta cantidad tiene la consideración de pago a cuenta de nuestro impuesto sobre la renta y funciona de una manera muy similar a las retenciones por el trabajo que se aplican en las nóminas. Esto significa, que desde el momento que nos damos de alta en Hacienda como empresarios, estamos obligados también a presentar la declaración de la renta, no pudiendo confirmar el borrador del IRPF.



## El IVA en las aplicaciones móviles

El IVA en las aplicaciones móviles se calcula en función del país de residencia del destinatario de la operación. Es decir, no se aplica el mismo IVA a un comprador español, que a un europeo que a un australiano. Las reglas que se siguen para calcular el IVA de cada venta son las siguientes:

- Si el comprador está en España, el IVA que cobraremos por cada aplicación será del 21%.
- Si el comprador está en Europa, en los países de la UE, y no es empresario, el IVA que se cobra es el 21%.
- Si el comprador es empresario y dispone de VAT (número fiscal europeo para el IVA), el IVA que cobraremos será el 0%.
- Si el comprador está fuera de la zona euro, el IVA que cobraremos será del 0%, ya sea empresario o no.

En el caso de Google Play, la propia plataforma puede discriminar la aplicación del IVA en función del país de compra y del tipo de comprador, por lo que este cálculo es sencillo y se puede configurar directamente en la tabla de precios de la aplicación.

Ahora, una vez que tenemos trimestralmente nuestras ventas, tendremos que ingresar trimestralmente la cantidad de IVA que hemos cobrado mediante el modelo 303 y también tendremos que confeccionar el modelo 349 a final de año siempre y cuando hayamos realizado ventas a empresarios europeos con IVA al cero.

## La Seguridad Social, aquí vienen los problemas:

Todo el proceso fiscal que hemos explicado, es un proceso farragoso, pero una vez que se mira todo con detalle, veremos que no es excesivamente difícil de realizar. El problema real lo tenemos con la Seguridad Social y la consideración de empresario. Según la Seguridad Social, un empresario es toda persona que realiza una actividad por cuenta propia (venta de aplicaciones móviles lo es) y de manera habitual (yo tengo a la venta las aplicaciones todos los días en Google Play).

Ser empresario en la Seguridad Social implica darse de alta como autónomo y pagar mensualmente las cotizaciones de este régimen que oscilan entre 190 y 250 euros al mes para los importes más bajos. Si yo ingreso con mi aplicación 100 euros al mes y me doy de alta como autónomo, perdería dinero por realizar estas ventas, por lo que tiene poca lógica tramitar este punto.

Para solventar este caso, existe una norma no escrita, fijada por una sentencia del Supremo del año 2007 que parte de la base que la habitualidad de la actividad empresarial debe correlacionarse con el nivel de ingresos que se obtienen mensualmente. El criterio general que se sigue para tramitar el alta como autónomo, es superar la barrera del salario mínimo interprofesional (640 euros/mes) por lo que a día de hoy, la Seguridad Social no está obligando al alta a aquellos “empresarios en situaciones especiales” que no ingresan mensualmente por encima de esta cuantía.

Pero como he dicho anteriormente, esta regla no está escrita en ninguna ley y en cualquier momento, la Seguridad Social podría sancionar a estos empresarios obligando al alta en autónomos desde la fecha del alta fiscal. Esta norma es equiparable a bloggers que venden su contenido a terceros o a los ingresos que se obtienen por publicidad en las webs y es un problema legal con una solución compleja que a día de hoy no la tiene.



## **2.3. Posibles ayudas o subvenciones para el desarrollo del proyecto**

### **2.3.1. Subvenciones**



#### **Medidas de Creación de Empleo y Fomento del Emprendimiento**

Con el Decreto-ley 8/2013, de 28 de mayo, se ponen en marcha herramientas para los jóvenes, los emprendedores y la economía social. (BOJA Nº 105, 31 de mayo de 2013)

#### **Programa de apoyo y fomento del trabajo autónomo**

**Convocatoria 2013:** CERRADA. Hasta 1/07/2013

**Convocatoria 2014:** PENDIENTE

**Órgano responsable:** Consejería de Economía, Innovación, Ciencia y Empleo.

#### **Dirigido a:**

Trabajadoras y trabajadores autónomos, las personas físicas que ejercen una actividad económica de forma habitual, personal, directa, por cuenta propia, dando o no ocupación a trabajadores por cuenta ajena y con residencia y domicilio fiscal en Andalucía.

#### **Objeto de la Subvención:**

Desarrollo de líneas de actuación dirigidas principalmente a crear más empresas y empleo, a consolidar y fortalecer el sector y al desarrollo de la cultura y la actividad emprendedora en el auto-empleo.

- **Línea 1. Creación de empleo en el trabajo autónomo.**

**Objeto de la Subvención:**

Contrataciones por tiempo indefinido ordinarias, hasta un máximo de tres, realizadas por cada persona trabajadora autónoma. Contratos de interinidad realizados en los supuestos de riesgo durante el embarazo y periodos de descanso por maternidad.

**Información complementaria:**

Las contrataciones deberán realizarse con personas jóvenes menores de treinta y cinco años, inclusive. Las personas que se contraten deberán estar desempleadas e inscritas como demandantes de empleo en el Servicio Andaluz de Empleo. Las contrataciones se deberán haber formalizado en el momento de la presentación de la solicitud.

**Cuantía de la Subvención:**

Un valor de 3.000 euros por cada nuevo contrato formalizado con carácter indefinido ordinario, cuando sea a jornada completa. Hasta 3.000 euros por un periodo de ocho meses cuando la contratación sea a jornada completa, por cada contrato de interinidad, para sustituir a personas con riesgo durante el embarazo. Hasta 1.500 euros por un periodo de dieciséis semanas cuando la contratación sea a jornada completa, por cada contrato de interinidad, para sustituir a personas que se encuentren en situación de permiso por maternidad.

- **Línea 2. Consolidación de empleo en el trabajo autónomo.**

**Objeto de la Subvención:**

Las transformaciones de contratos de duración determinada en contratos de tiempo indefinido ordinario, hasta un máximo de tres, realizadas por cada persona trabajadora autónoma.

**Cuantía de la Subvención:**

Un valor de 3.000 euros por cada nueva transformación de contrato de duración determinada en contratos de tiempo indefinido ordinario, cuando sea a jornada completa.

- **Línea 3. Creación de empresas de trabajo autónomo.**

**Objeto de la Subvención:**

Subvenciones para el inicio de la actividad, con medidas destinadas a personas que se establezcan como trabajadoras autónomas, bien sea por primera vez o que en los últimos cinco años no hayan estado dadas de alta en el



Régimen Especial de Trabajadores Autónomos y personas que han capitalizado la prestación por desempleo en su modalidad de pago único.

### **Cuantía de la Subvención:**

Un valor de 2.000 euros para la medida destinada a personas que se establecen como trabajadoras autónomas y el importe equivalente al 50% de las cuotas para la medida destinada a personas que han capitalizado la prestación por desempleo en su modalidad de pago único.

- **Línea 4. Consolidación empresarial del trabajo autónomo.**

### **Objeto de la Subvención:**

Desarrollar y fortalecer el tejido empresarial andaluz del trabajo autónomo, mediante la Cooperación empresarial, Relevo generacional de las unidades económicas de trabajo autónomo, innovación empresarial y Cohesión y competitividad empresarial en el trabajo autónomo.

### **Conceptos subvencionables:**

Creación de una empresa, desarrollo de acciones de dinamización que incidan en proyectos de cooperación, acciones de sensibilización, asesoramiento, capacitación y autorización, estudios dirigidos al diseño y elaboración de planes sectoriales, capacitación y asesoramiento técnico, gastos de personal, gastos de desplazamiento y estancias, gastos generales para la ejecución del proyecto, gastos de contratación de servicios especializados externos.

### **Cuantía de la Subvención:**

Para la medida de cooperación empresarial destinada a la constitución de una empresa, hasta el 100% de los gastos de constitución, con un tope máximo de 1.000 euros.

Para el resto de las medidas, hasta el 100% de gastos subvencionables necesarios para la ejecución de cada una de ellas, con un límite de 90.000 euros, en la siguiente proporción: hasta un 80% de gastos de personal y desplazamiento y hasta un 20% de gastos generales.

- **Línea 5. Fomento de la innovación en el trabajo autónomo.**

### **Objeto de la Subvención:**

Implantación y el desarrollo de proyectos innovadores que se promuevan por las personas beneficiarias para su establecimiento como persona trabajadora por cuenta propia o para la consolidación de la actividad económica.



**Gastos subvencionables:**

Gastos por inicio de la actividad, contratación por tiempo indefinido ordinario y la transformación de contratos de duración determinada en contratos por tiempo indefinido ordinario, bienes de equipo, equipos informáticos o de infraestructura de TIC en general, adquisición y tratamiento de software, activos fijos inmateriales, diseño de productos, envases y embalajes, estrategias de comunicación y de imagen de la empresa, certificación de sistemas de gestión...

**Cuantía de la Subvención:**

Un valor de 2.000 euros para aquellas personas beneficiarias que, mediante la implantación y desarrollo de un proyecto innovador, hayan creado una unidad económica de trabajo autónomo o se integren en una sociedad civil o comunidad de bienes.

Hasta 8.000 euros para la implantación y desarrollo del proyecto de innovación, en función de la inversión prevista presentada.

3.000 euros por cada nueva contratación por tiempo indefinido ordinario o transformaciones de contratos de duración determinada en contratos por tiempo indefinido ordinario, a jornada completa y hasta un máximo de tres contrataciones indefinidas o transformaciones de contratos.

- **Línea 6. Promoción del trabajo autónomo.**

**Objeto de la Subvención:**

Promover el conocimiento y el desarrollo del trabajo autónomo. Conceptos subvencionables: acciones para la mejora del conocimiento entre la población del trabajo autónomo, acciones de información, motivación y orientación, acciones de asesoramiento individualizado, acciones de autorización, acciones de asistencia técnica...

**Cuantía de la Subvención:**

Hasta el 100% de gastos subvencionables necesarios para la ejecución de cada una de las medidas, en una proporción de hasta un 80% de gastos de personal y desplazamiento y un 20% de gastos generales.

### 2.3.2. Ayudas

**Programa de incentivos a la innovación y al desarrollo empresarial.**

**Plazo solicitud convocatoria 2014:** Desde 1/07/2013 hasta 30/04/2014

**Organismo:** Consejería de Economía, Innovación y Ciencia.

**Proyectos subvencionables:**

- Proyectos que contribuyan a la innovación, investigación y desarrollo, y a la cooperación empresarial y al crecimiento y consolidación de las empresas andaluzas.
- Proyectos que tengan como finalidad la creación de una nueva empresa.
- Proyectos que tiendan a la modernización de empresas y para ello incorporen mejoras sustanciales en los productos, procesos o servicios, organización o modelo de negocio.
- Proyectos que promuevan la innovación, la competitividad y la productividad de las empresas mediante actuaciones basadas en la cooperación entre empresas.
- Proyectos que tengan por objeto la generación de un conocimiento nuevo o tecnologías nuevas.
- Proyectos que apliquen un conocimiento existente para la obtención de nuevos desarrollos.
- Proyectos desarrollo tecnológico experimental para la obtención de nuevos desarrollos.
- Proyectos altamente generadores de empleo.

**Cuantía máxima:** Hasta un 40% en función de la línea de ayudas.

### 3. ANALISIS DEL SISTEMA ACTUAL

Dado que es una aplicación que no se integrara o complementará a ninguna aplicación existente, no se realizara análisis del sistema actual.

### 4. SOLUCION PROPUESTA

#### 4.1. *Diferentes tecnologías existentes para el desarrollo y puesta a punto de la aplicación que se desarrollara.*

##### 4.1.1. Comparativa de los distintos sistemas operativos para dispositivos móviles

###### ➤ IOS (Apple)

El sistema operativo de Apple, donde se cuida hasta el más mínimo detalle y se mantiene un estricto control sobre el hardware y el software. La integración de las aplicaciones con el hardware es perfecta, debido a que solo los dispositivos de Apple pueden contar con este sistema operativo. Es uno de los sistemas con más aplicaciones, y normalmente el primero en recibir las nuevas creaciones. La integración con todo el ecosistema de Apple es, simplemente, perfecta.



Entre las desventajas encontramos la poca posibilidad de personalización del terminal, y que se trate de un sistema totalmente cerrado. Las aplicaciones solo pueden obtenerse desde la App Store, que impone unas férreas medidas de control. El intercambio de ficheros con dispositivos de otras compañías es prácticamente imposible.

Uno de los recursos más interesantes del iPhone es el Facetime, que permite video llamadas de un iPhone a otro o desde un iPhone a un Mac. Actualmente los Smartphone con Android dependen de las aplicaciones del otro equipo para poder funcionar.

Los usuarios de iPhone siempre cuentan con la versión más reciente del

## Desarrollo de aplicaciones multiplataforma

sistema operativo gracias a una constante actualización automática. En el caso de Android, solamente existe la fragmentación del sistema y la actualización depende exclusivamente del fabricante del equipo.

### ➤ Android

Es el sistema operativo más utilizado del planeta, y es el que está experimentando un mayor crecimiento. Una de las ventajas es que es un sistema de código abierto y permite que cualquiera pueda modificar y mejorarlo, sin ningún tipo de restricción. La comunidad de desarrolladores es muy activa y está creando continuamente soluciones para Android. El sistema se puede utilizar en casi cualquier Smartphone, lo que ayuda a que su presencia sea mayoritaria en el planeta.



Una de las grandes ventajas es la posibilidad de elegir entre distintos equipos con Android, ya que son cada vez más los disponibles en el mercado. El iPhone está solo, limitando la elección.

Ser el sistema más utilizado también tiene sus desventajas. En primer lugar, es el foco de casi todos los ataques de malware. Las actualizaciones suelen tardar demasiado en llegar a los usuarios finales, debido a que deben pasar primero por el fabricante. La duración de la batería también se ve afectada por las aplicaciones que estén corriendo en segundo plano.

Otra ventaja de Android es la posibilidad de reproducir videos en alta definición en alguna pantalla o televisor, gracias a que varios modelos poseen puertos HDMI. El iPhone, además de no poseer este puerto, tampoco tiene slot para tarjetas de memoria, cosa que la mayoría de los equipos con Android disponen para poder expandir su capacidad con el simple hecho de añadir una tarjeta micro SD.

### ➤ Windows Phone

## Desarrollo de aplicaciones multiplataforma

Windows Phone es un sistema operativo móvil desarrollado por Microsoft como sucesor de Windows Mobile. A diferencia de su predecesor está enfocado en el mercado de consumo en lugar de en el mercado empresarial. Con Windows Phone Microsoft ofrece una nueva interfaz de usuario que integra varios de sus servicios propios como OneDrive, Skype y Xbox Live en el sistema operativo.



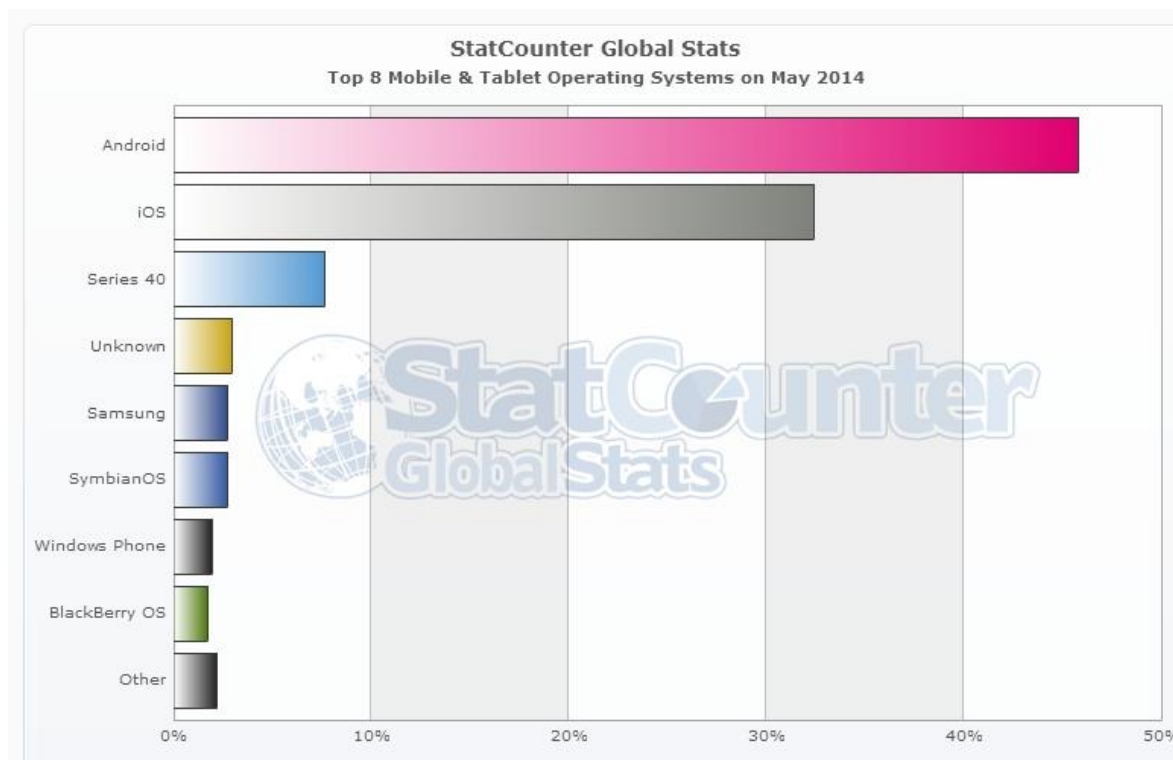
Microsoft trabaja de cerca con los fabricantes y se nota, Windows Phone 8 es realmente fluido. La transición entre pantallas es intuitiva y el teclado funciona sin problemas. En general, creo que nadie puede negarlo: La interfaz es diferente, atractiva e innovadora respecto a sus competidores. Y hay que dar crédito a Microsoft por crear el primer ecosistema con una experiencia de uso similar en las tres pantallas: PC, móvil y tableta. Tiene sus fallos y llegan tarde al móvil, pero su apuesta es coherente.

Falla el ecosistema de apps. Es la mayor desventaja. Microsoft asegura que hay 120.000 aplicaciones listas, pero eso es añadiendo las de Windows Phone 7.5, que son compatibles. Los desarrolladores están por la labor de volcarse en Windows Phone 8 por lo que debería ser cuestión de tiempo ver apps en iOS, Android y WP8 casi por defecto.

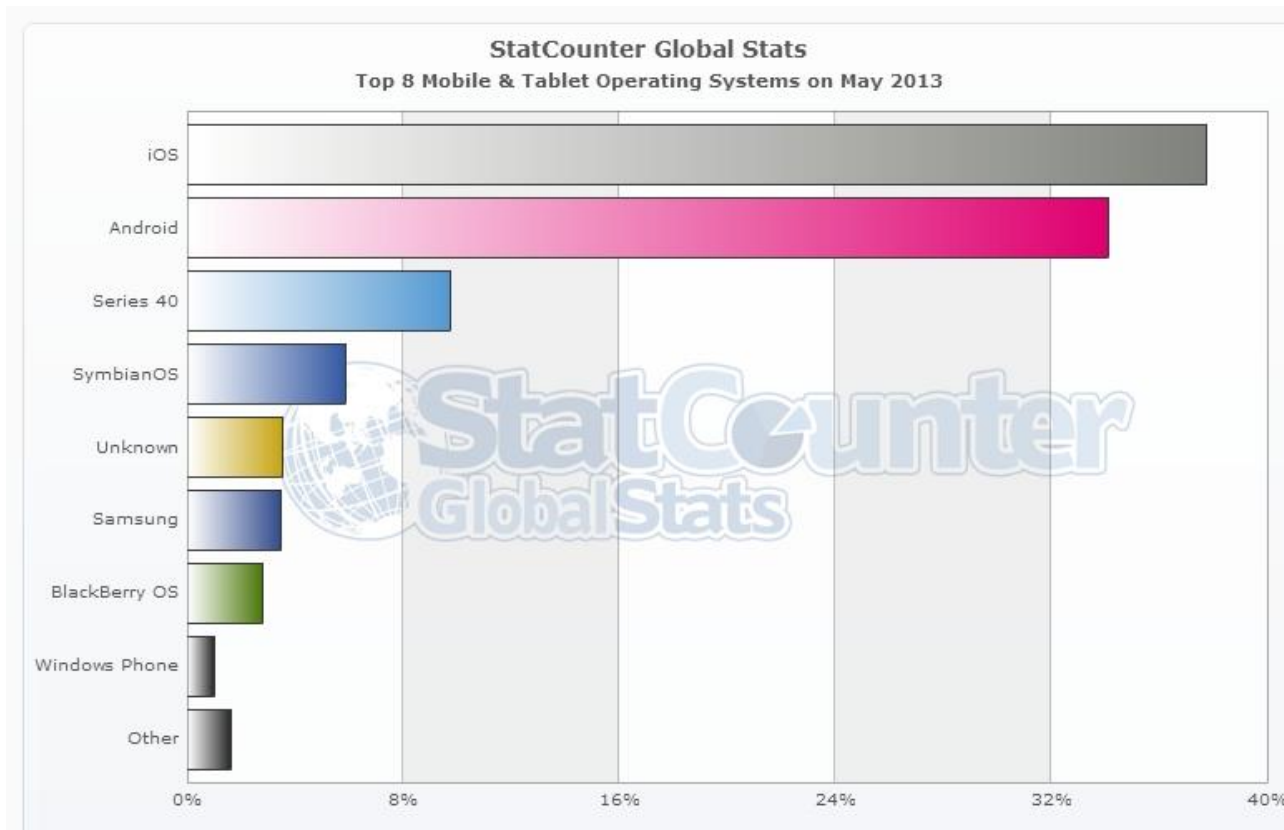
#### 4.1.2. Gráficas de estadísticas

A continuación vamos a mostrar unas gráficas de estadísticas de los distintos porcentajes de sistemas operativos tanto para móviles como para escritorio en todo el mundo con fecha de Mayo de 2014:

- Sistemas Operativos para Móviles.



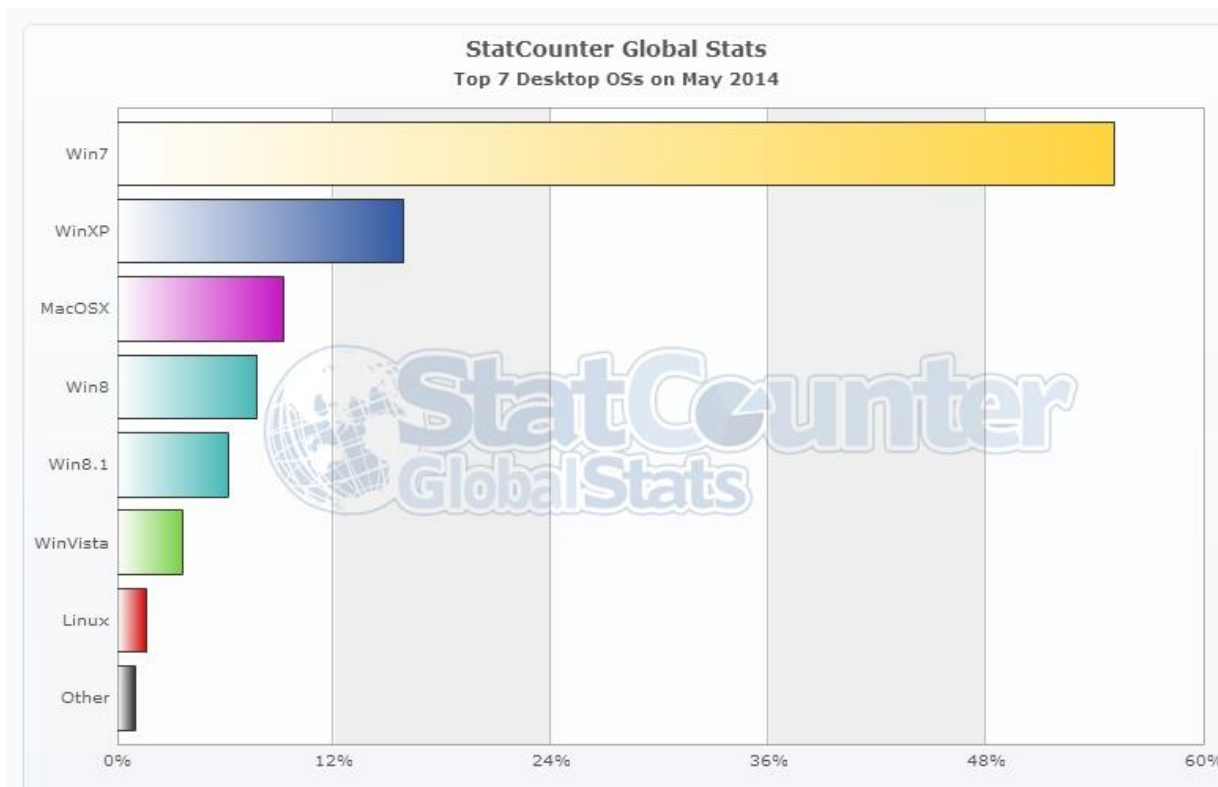
Como podemos observar el mercado está totalmente conquistado por Android e iOS con una ventaja bastante amplia para Android, la cual si comparamos con fechas anteriores podemos apreciar cómo ha aumentado bastante:



Podemos apreciar en esta segunda gráfica, como en tan solo un año el sistema de Android ha alcanzado el primer puesto con un aumento del 35% aproximadamente al 45% que tiene en la actualidad, iOS delegado a una segunda posición por el gran aumento de Android.

También podemos apreciar como el resto de sistemas ha disminuido ligeramente en beneficio de Android.

- Sistemas Operativos de Escritorio.



Respecto a los Sistemas Operativos de escritorio apreciamos que el mercado es de Microsoft casi en su totalidad, ya que Mac siempre tiene su porcentaje de clientes fieles.

Con esto podemos valorar debido a la integración de la Tienda en las últimas versiones de Windows como un nuevo mercado en el cual sacar partido con nuestras aplicaciones.



### 4.1.3. Introducción a libGDX

libGDX es un framework multiplataforma para el desarrollo de juegos. Actualmente soporta las plataformas Windows, Linux, Mac OS X, Android, iOS y HTML5.



Permite escribir el código una vez y llevarlo a varias plataformas sin hacer modificaciones. Una gran ventaja es que se pueden realizar iteraciones más rápidas al poder testear la aplicación en desktop. Permite ir lo bajo nivel que uno desee, dando acceso directo al sistema de archivos, dispositivos de entrada, de audio y a Open GL a través de una interfaz unificada.

Posee un grupo de APIs muy poderosas que ayudan en tareas de desarrollo de juegos como por ejemplo renderizar Sprites, texto, crear interfaces de usuario, reproducir efectos de sonido y música, realizar cálculos algebraicos y trigonométricos, parsear a JSON y XML, entre otras.

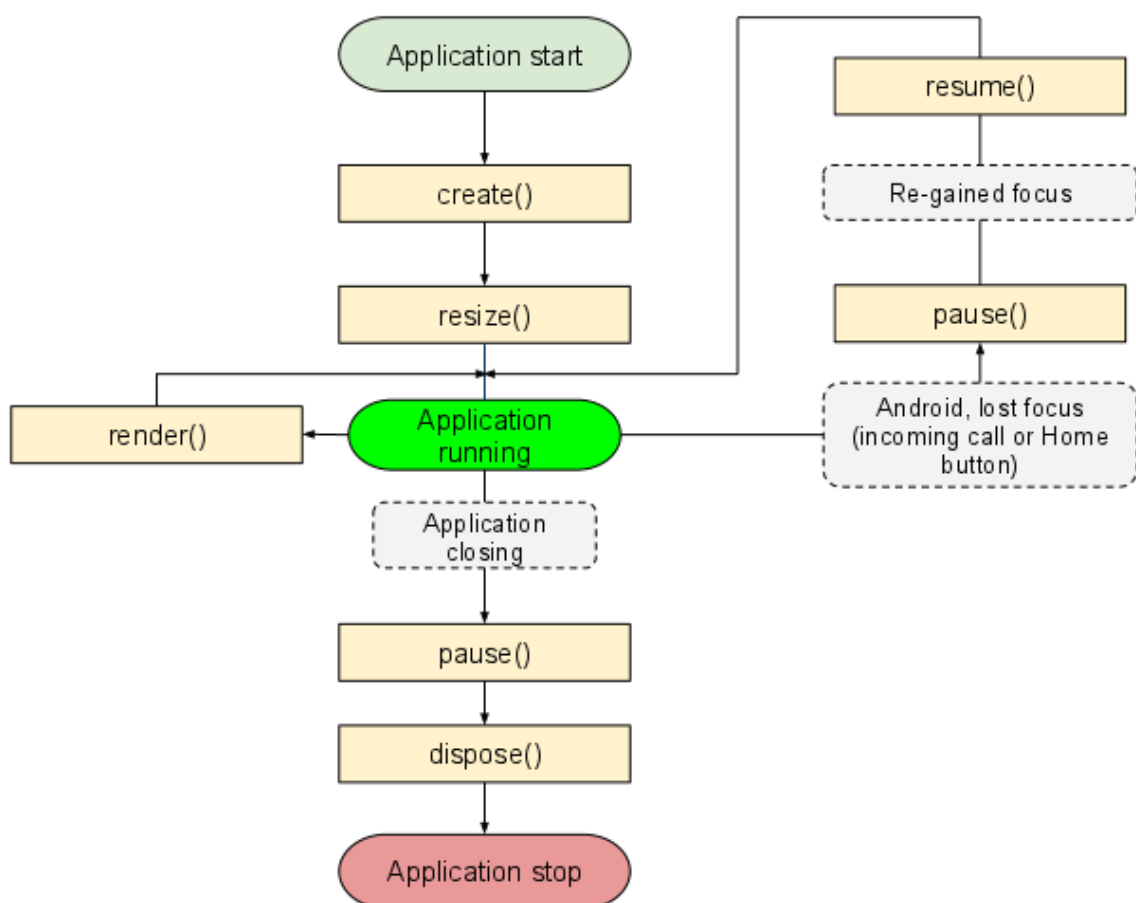
libGDX emplea librerías de terceros para proveer mucha de su funcionalidad, algunos ejemplos que nos encontramos:

- **Open GL** (Para escribir aplicaciones que produzcan gráficos 2D y 3D)
- **LWJGL** ( Lightweight Java Game Library)
- **MPG123** (reproductor/decodificador de audio)
- **Freetype** (Librería para renderizar fuentes con alta calidad)
- **Box2D** (Motor para físicas en dos dimensiones, usado por Angry Birds)
- **RoboVM** (Librería para exportar proyectos Java a la plataforma iOS)
- **GWT** (Google Web Toolkit, herramienta para compilar proyectos nativos en HTML5 y Javascript)

#### Ciclo de vida

- **Create():** se llama cuando la aplicación se crea por primera vez

- **Resize():** Se llama cuando al aplicación se redimensiona. Puede ocurrir en cualquier momento en el que la aplicación este corriendo pero nunca ocurrirá antes de llamar al método create(). El evento solo es posible en la aplicación de escritorio
- **Pause():** Se llama cuando la aplicación se pausa. Una aplicación es pausada antes de que sea destruida, cuando un usuario presionó el botón Home de Android o ha recibido una llamada. En escritorio solo será llamado inmediatamente antes de que el método dispose() sea llamado.
- **Dispose():** Llamado cuando la aplicación es destruida. Precedido por una llamada a pause().
- **Render():** Se llama cuando la pantalla debe renderizarse a sí misma.
- **Resume():** Se llama cuando la aplicación vuelve del estado pause(). En Android ocurre cuando la aplicación obtiene el foco de nuevo. En escritorio este método nunca será llamado.



## Application, StarterClass y ApplicationListener

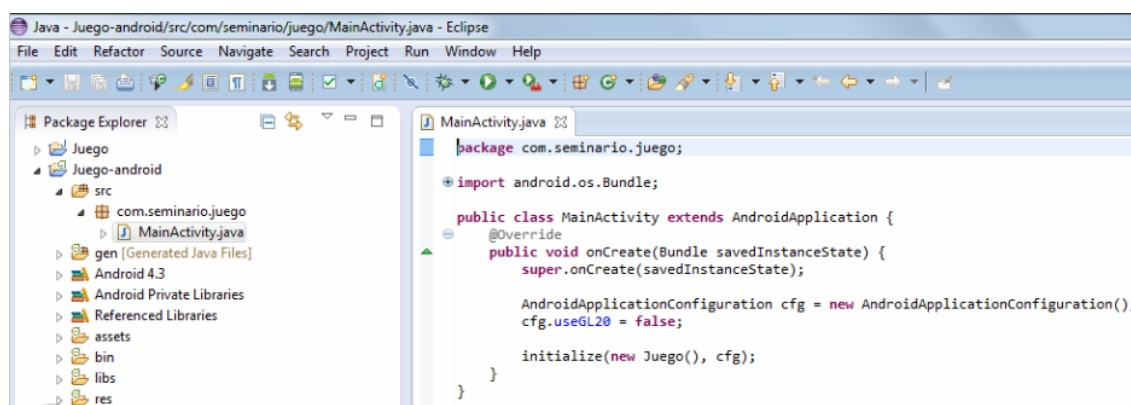
Application es el proyecto core, o sea el punto de entrada principal del proyecto. Setea una ventana y una superficie de renderización y maneja los distintos aspectos de la aplicación: Gráficos, audio, inputs y archivos.

Al haber creado los proyectos para Android, Desktop, iOS y HTML5, tendremos varias implementaciones de esta interfaz:

- AndroidApplication (Android)
- LwjglApplication (Escritorio)
- IOSApplication.Delegate (iOS)
- GwtApplication (HTML5)

La StarterClass es el único código que necesita ser escrito para cada plataforma específica es el de las "starter classes". Para cada plataforma, una porción de código dentro de estas clases instanciará una implementación concreta de la interfaz "Application".

Al crear el proyecto con la GUI que nos brinda LibGDX, estas clases serán creadas automáticamente con código por defecto que puede ser modificado. Ejemplo de proyectos para Android y Desktop:



```
Java - Juego-android/src/com/seminario/juego/MainActivity.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help

Package Explorer
  Juego
    Juego-android
      src
        com.seminario.juego
          MainActivity.java
        gen [Generated Java Files]
        Android 4.3
        Android Private Libraries
        Referenced Libraries
        assets
        bin
        libs
        res

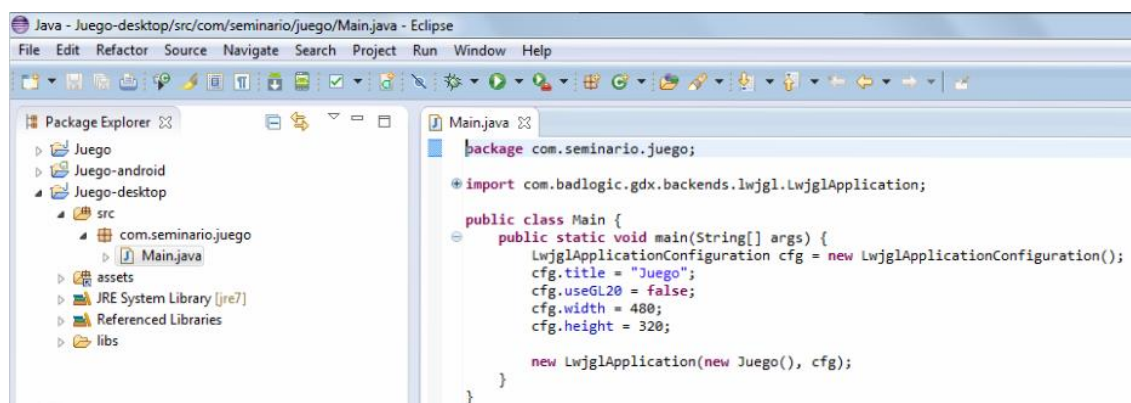
MainActivity.java
package com.seminario.juego;

import android.os.Bundle;

public class MainActivity extends AndroidApplication {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        AndroidApplicationConfiguration cfg = new AndroidApplicationConfiguration();
        cfg.useGL20 = false;

        initialize(new Juego(), cfg);
    }
}
```



```
Java - Juego-desktop/src/com/seminario/juego/Main.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help

Package Explorer
  Juego
    Juego-android
    Juego-desktop
      src
        com.seminario.juego
          Main.java
        assets
        JRE System Library [jre7]
        Referenced Libraries
        libs

Main.java
package com.seminario.juego;

import com.badlogic.gdx.backends.lwjgl.LwjglApplication;

public class Main {
    public static void main(String[] args) {
        LwjglApplicationConfiguration cfg = new LwjglApplicationConfiguration();
        cfg.title = "Juego";
        cfg.useGL20 = false;
        cfg.width = 480;
        cfg.height = 320;

        new LwjglApplication(new Juego(), cfg);
    }
}
```

## Desarrollo de aplicaciones multiplataforma

En el método inicializador de cada starter class, se especifica una configuración, y se crea luego una instancia de la aplicación. Para ello se pasan dos parámetros

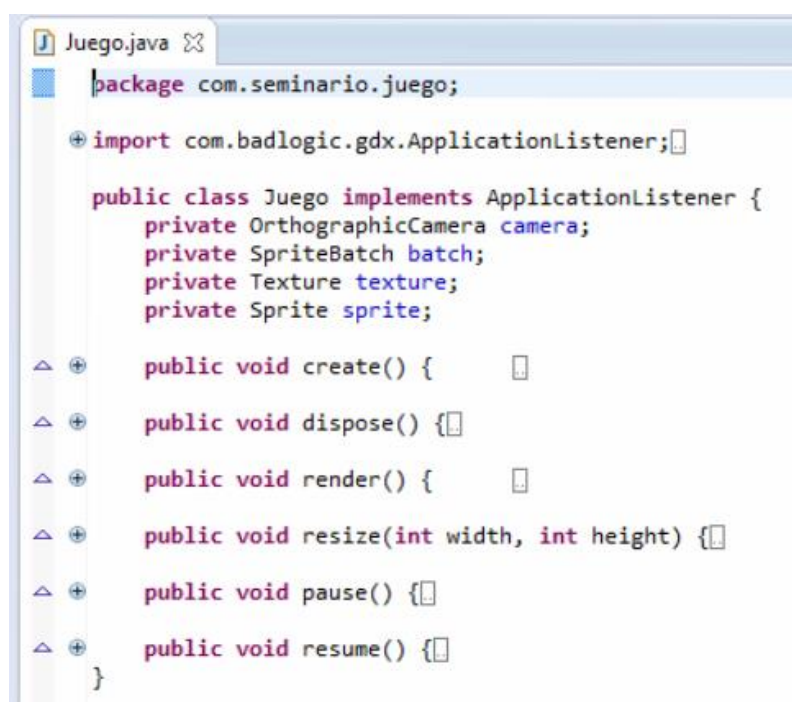
1. Una nueva instancia de la implementación de ApplicationListener que se encuentra en el proyecto del core. (En el proyecto se llama Juego.java).
2. La configuración especificada.

La aplicación luego llamará a los métodos del "ApplicationListener" en los momentos apropiados.

En la implementación de la interfaz ApplicationListener se le da comportamiento a los eventos del ciclo de vida (create(), resize(), render(), pause(), dispose() y resume()), y se le pasa una instancia de esa implementación a la Aplicación de cada una de las plataformas para las que se desee desarrollar el juego.

La aplicación la llamará cada vez que un evento ocurra.

En esta imagen se puede ver la implementación de ApplicationListener que se crea por defecto al crear un proyecto:



```
Juego.java
package com.seminario.juego;

import com.badlogic.gdx.ApplicationListener;

public class Juego implements ApplicationListener {
    private OrthographicCamera camera;
    private SpriteBatch batch;
    private Texture texture;
    private Sprite sprite;

    public void create() { }
    public void dispose() {}
    public void render() { }
    public void resize(int width, int height) {}
    public void pause() {}
    public void resume() {}
}
```

LibGDX está compuesto por cuatro módulos, a los cuales se accede por medio de la clase estática Gdx:

- **Input:** Gestiona la entrada a través de teclado, pantalla táctil, acelerómetro, etc.

- **Graphics:** Permite gestionar la representación de imágenes en la pantalla.
- **Files:** Se trata de un componente para lectura y escritura de ficheros de datos como imágenes, archivos de configuración, sonidos, música, texturas, etc.
- **Audio:** Facilita la reproducción y grabación de audio en todas las plataformas que soporta.

### Input

En desktop, los usuarios se pueden comunicar con la aplicación por medio del teclado y el mouse. En Android, el mouse es reemplazado por una pantalla táctil, y puede existir un teclado físico. Todos los dispositivos compatibles con Android también poseen acelerómetro y algunos un compass (sensor de campo magnético).

LibGDX abstrae todos esos dispositivos de entrada. El mouse y las pantallas táctiles son tratadas como si fueran lo mismo.

Para la entrada de datos por parte del usuario se puede hacer polling o registrar un listener que recibirá eventos de entrada en orden cronológico.

Todas las facilidades de entrada son accedidas por medio del módulo Input de LibGDX.

Ejemplo:

```
if (Gdx.input.isKeyPressed(Keys.BACK) || Gdx.input.isKeyPressed(Keys.ESCAPE)) {  
    juego.setScreen(new MenuPrincipal(juego));  
}
```

### Graphics

El módulo Graphics provee información acerca del display del dispositivo actual y la ventana de la aplicación como información y acceso al contexto de OpenGL.

Como el resto de los módulos, se accede a través de los campos estáticos de la clase Gdx.

Uno de los métodos más útiles en la clase Graphics es `getDeltaTime()`, el cual provee el tiempo transcurrido desde el último cuadro renderizado. Esto puede ser útil para animaciones basadas en tiempo.

Otro método útil es `getFramesPerSecond()`, que retorna un promedio del rango de cuadros para propósitos de diagnóstico.

## Texture, TextureRegion y SpriteBatch

### Texture

Una imagen que es subida al GPU (unidad de procesamiento de graficos) es denominada una textura (Texture). Las medidas de las imágenes deben ser potencias de dos, salvo que se trate de una aplicación que utiliza OpenGL 2.0.

```
private Texture texturaMoneda;  
texturaMoneda = new Texture(Gdx.files.internal("moneda.png"));
```

### TextureRegion

Una forma que representa una porción de una textura es denominada una región de textura (TextureRegion):

```
private TextureRegion regionMoneda;  
regionMoneda = new TextureRegion(texturaMoneda, 0, 0, 32, 32);
```

En el ejemplo anterior, se crea una región de textura de 32 x 32 píxeles, partiendo desde el punto 0,0 de la textura.

### SpriteBatch

Es común dibujar una textura mapeada a un rectángulo, y dibujar la misma textura o una región de la textura varias veces. Sería ineficiente mandar cada rectángulo al GPU para que sea dibujado. Por eso, muchos rectángulos para la misma textura pueden ser descriptos y mandados al GPU al mismo tiempo por medio de la clase SpriteBatch.

A este se le manda una textura y las coordenadas de cada rectángulo. Este recolecta la forma geométrica sin enviársela al GPU.

Todas las llamadas para dibujar con SpriteBatch deben ser realizadas entre sus métodos begin() y end(). Esto se puede ver en el siguiente fragmento de código del juego:

```
public void renderizarObjetos() {  
    getSpriteBatch().enableBlending();  
    getSpriteBatch().begin();  
    renderizarMeta();  
    renderizarNinja();  
    renderizarPlataformas();  
    renderizarItems();  
    renderizarEnemigos();  
    renderizarShurikens();  
    getSpriteBatch().end();  
}
```

Para dibujar texturas utilizando un SpriteBatch, se utiliza su método `draw()`. Posee varias sobrecargas para dibujar texturas y regiones de texturas.

### Files

El modulo Files de libGDX permite:

- Leer un archivo.
- Escribir en un archivo.
- Copiar un archivo.
- Mover un archivo.
- Eliminar un archivo.
- Listar archivos y directorios.
- Averiguar si un archivo o directorio existe.

Cada una de las plataformas maneja el I/O de archivos de una forma diferente, pero gracias a este módulo el programador no tiene que preocuparse por ello.

### Tipos de archivos

Un archivo en libGDX es representado por una instancia de la clase `FileHandle`. Esta posee un tipo, que define en donde se encuentra el archivo:

- **Classpath:** Almacenados en las carpetas fuentes. Son empaquetados con los jars y siempre son de solo lectura. Deben ser evitados si es posible.
- **Internal:** Se encuentran en la carpeta `assets` del proyecto de Android que está linkeada al resto de los proyectos. Sólo se pueden leer archivos.
- **Local:** Almacenamiento interno. En escritorio los archivos están juntos al JAR o en la carpeta del proyecto. En Android están en el almacenamiento interno del teléfono.
- **External:** Relativos a la raíz de la tarjeta SD en Android, y en la carpeta personal del usuario en desktop. Se recomienda su uso para datos volátiles, es decir, que no son imprescindibles para la aplicación.
- **Absolute:** Necesitan tener sus rutas completas especificadas. Por temas de portabilidad esta opción debe ser solo utilizada cuando sea absolutamente necesaria.



Existen cinco métodos en Gdx.files:

- Gdx.files.classpath(String): FileHandle
- Gdx.files.internal(String): FileHandle
- Gdx.files.local(String): FileHandle
- Gdx.files.external(String): FileHandle
- Gdx.files.absolute(String): FileHandle

(String: ruta del archivo)

Se pueden tener Fileandles de archivos que no existan. Si se intenta escribir sobre ellos, se crearán automáticamente. Si se intenta leer sobre ellos dará un error.

Por ejemplo, para crear una textura a partir de un archivo que se encuentra en la carpeta assets (internal):

```
Texture texturaFondo = new Texture(Gdx.files.internal("fondo_menu.png"));
```

En este otro ejemplo del juego se obtiene un FileHandle que representa un archivo externo donde están guardados los datos del juego:

```
FileHandle fh = Gdx.files.external("ninjump.json");
```

Algunos métodos de lectura:

- readString()
- readBytes()

Algunos métodos de escritura:

- writeString(String, boolean);
- writeBytes(byte[], boolean);

### Audio

LibGDX provee métodos para reproducir efectos de sonidos cortos, o música desde el disco. También provee acceso al hardware de audio. El acceso a las facilidades de audio que brinda se hace por medio del módulo audio.

LibGDX pausa el audio cuando la aplicación está pausada y vuelve a reproducirlo cuando se regresa a la aplicación.



## Efectos de sonido

Los efectos de sonido son pequeños archivos de audio de unos pocos segundos, LibGDX soporta formatos MP3, OGG y WAV. Estos están representados por la interfaz Sound. Para cargar un efecto de sonido:

```
Sound sonido = Gdx.audio.newSound(Gdx.files.internal("miSonido.mp3"));
```

El código anterior carga un archivo de audio llamado "miSonido.mp3" de la carpeta assets.

El siguiente método reproduce el sonido una vez, con el volumen al máximo. Retorna un long con el id de la instancia de sonido que se está reproduciendo.

```
long id = sonido.play(1.0f);
```

El método stop detiene el sonido, recibiendo como parámetro el id de la instancia.

```
sonido.stop(id);
```

Sound posee métodos para el procesamiento del audio, como por ejemplo:

- `setPitch()`: cambia el tono del sonido.
- `setPan()`: cambia el paneo del sonido, siendo -1 la izquierda y 1 la derecha.
- `setLooping()`: hace que el sonido se reproduzca como un bucle.

Una vez que no se necesite, hay que asegurarse de desecharlo, y de no acceder a él una vez desechado ya que provocará errores:

```
sonido.dispose();
```

## Música

Para cualquier sonido que sea más largo que varios segundos es preferible reproducirlo desde el disco que cargarlo completamente en la RAM. Para ello LibGDX provee la interfaz Music.

Para cargar una instancia de Music:

```
Music musica = Gdx.audio.newMusic(Gdx.files.internal("musica.mp3"));
```

Para reproducir la música se utiliza el método `play()`:

```
musica.play();
```

También posee otros métodos para el procesamiento de la música al igual que Sound:

```
musica.setVolume(0.5f);
```

```
musica.setLooping(true);  
musica.stop();  
musica.pause();  
musica.play();  
boolean isPlaying = musica.isPlaying();  
boolean isLooping = musica.isLooping();  
float position = musica.getPosition();
```

Las instancias de Music son pesadas, por lo que se deben desechar si dejan de ser necesitadas para liberar recursos:

**musica.dispose();**

## Scene2D

Scene2D se utiliza para crear aplicaciones e interfaces de usuario utilizando una jerarquía de actores. Provee las siguientes funcionalidades:

- La rotación y la escala de un grupo son aplicadas a todos los actores hijos.
- Dibujo simplificado a través de SpriteBatch.
- Detección de colisiones de actores. Cada uno determina si colisionó.
- Ruteo de eventos de entrada u otros eventos al actor apropiado.
- Sistema de acciones para la manipulación fácil de actores en el tiempo.

Scene2D está bien equipado para crear menús de juegos, HUDS, herramientas, y otras interfaces de usuario. El paquete scene2d.ui provee varios actores y otras utilidades específicamente para la creación de interfaces de usuario.

Los actores almacenan información que es considerada parte del modelo pero también son la vista, ya que saben cómo dibujarse. Esto hace que la separación MVC sea difícil. Cuando es utilizado sólo para interfaces de usuario o aplicaciones que no utilizan MVC, esto no es un problema.

Scene2D posee tres clases en su núcleo.

- La clase Actor, con una posición, medida rectangular, punto de origen, escala, rotación y color.
- La clase Group, que es un actor que puede tener otros actores hijos.
- La clase Stage posee una cámara, un SpriteBatch, y un grupo. Administra el dibujo de los actores y distribuye los eventos de entrada.

Cuando se llama al método draw del Stage, este llama al draw de todos sus actores. Lo mismo sucede cuando se llama a su método act.

El método draw de un actor dibuja una región utilizando la información del actor. El SpriteBatch que se le pasa para dibujar es configurado para dibujar en las coordenadas del padre, por lo que 0,0 es el borde inferior izquierdo del padre. Esto hace que dibujar sea simple, por más que el padre esté rotado y escalado.

## InputListener

Para la mayoría de tipos de eventos, se utilizan listeners específicos por conveniencia. Por ejemplo, InputListener se utiliza para recibir y manejar eventos de entrada. Un actor sólo necesita agregar un InputListener para comenzar a recibir eventos de entrada. Esta clase posee varios métodos que pueden ser sobrescritos:

```
botonJugar = new TextButton(" jugar ", estiloBotones);  
botonJugar.addListener(new InputListener() {  
    public boolean touchDown(InputEvent event, float x, float y,int pointer, int  
button) {  
        juego.reproducirSonido(sonidoBoton);  
        juego.setScreen(new SeleccionNivel(juego));  
        return true;  
    };  
});
```

En el ejemplo se le registra un InputListener a un TextButton (widget de Scene2d). En el listener se sobrescribe el método touchUp, evento propio del mouse o touchscreen. También se pueden sobrescribir otros métodos para estos dispositivos de entrada, como touchDragged, touchUp, enter, exit, mouseMoved, entre otros. También se pueden manejar eventos de teclado, como keyDown, keyUp, y keyTyped.

## SCENE2D.UI

Para la creación de interfaces de usuario, el paquete scene2d.ui brinda widgets y otras clases creadas sobre scene2d.

## Widget y Widgetgroup

Las clases Widget y WidgetGroup extienden de Actor y Group respectivamente. Los widgets de interfaz de usuario deberán extender a WidgetGroup si poseen actores hijos. Si no, de Widget.

## Layout

Los widgets de UI no setean su medida y posición. Es el widget padre quien setea la medidas y posición de sus hijos. Los Widgets proveen medidas mínimas, preferidas, y máximas, que el padre usar como “pistas”. Existen muchas clases para la creación de UI utilizando scene2d. Algunas son:

- **Table:** Es un WidgetGroup que setea la medida y posición de sus hijos utilizando una tabla lógica, como las blas HTML.
- **Label:** Muestra texto utilizando un BitmapFont y un color.
- **Image:** Muestra un Drawable, que puede ser una textura, una región,un sprite, etc.

- **Button:** Se trata de un botón vacío. Al extender de Table, se le pueden añadir otros widgets.
- **TextButton:** Extiende de Button y contiene un label.
- **ScrollPane:** Permite scrollear un widget hijo con scrollbars y/o arrastrando el mouse o el dedo.

Existen muchos otros widgets como por ejemplo ImageButton, CheckBox, ButtonGroup, TextField, Stack, List, SelectBox, Slider, SplitPane, Window, Touchpad, Tree, Dialog, entre otros.

### Asset Manager

Es recomendado utilizar un AssetManager para administrar los assets de la aplicación. Aunque si es muy simple y si la carga de recursos no tarda mucho tiempo no es necesario.

Las ventajas que brinda el uso de la clase AssetManager son las siguientes: La carga de la mayoría de los recursos se hace de manera asíncrona, por lo que se puede mostrar una pantalla que reaccione a la carga de los assets.

Si existen por ejemplo dos assets A y B que dependen de un asset C, C no será desechado hasta que A, B, y C sean desechados. También si se carga un asset muchas veces, ocupará memoria sólo una vez.

Se tiene un solo lugar donde se pueden almacenar todos los assets. Para crear un AssetManager es muy simple:

```
AssetManager adminAssets = new AssetManager();
```

Para cargar los assets primero se debe llamar al método **load()**:

```
adminAssets.load("introduccion.png", Texture.class);  
adminAssets.load("botones.atlas", TextureAtlas.class);  
adminAssets.load("fuente_negra.fnt", BitmapFont.class);  
adminAssets.load("botonMenu.mp3", Sound.class);  
adminAssets.load("musicaMenu.mp3", Music.class);
```

Esas llamadas pondrán en una cola a los assets para ser cargados. Para cargarlos, se debe llamar al método **update()** continuamente: (**getAdminAssets** devuelve la instancia de **AssetManager**).

```
public void render(float delta) {  
    if (juego.getAdminAssets().update())  
        ((Game) Gdx.app.getApplicationListener()).setScreen(new  
        MenuPrincipal(juego));  
}
```

Mientras que el método `update()` del `AssetManager` retorne `false`, seguirá cargando assets. Para consultar el porcentaje de carga se puede utilizar el método `getProgress()`.

Para utilizar los assets cargados en el `AssetManager`:

```
juego.getAdminAssets().get("ayuda1.png", Texture.class);
```

Esto es, asumiendo que los assets fueron cargados. Si se desea consultar si un asset se cargó se puede hacer lo siguiente:

```
if (juego.getAdminAssets().isLoading("ayuda1.png")){  
    //hacer algo...  
}
```

También es simple desechar los assets utilizando el `AssetManager`:

```
juego.getAdminAssets().unload("ayuda1.png");
```

Es importante mencionar que los assets administrados por el `AssetManager` no deberían desecharse manualmente, sino con el método **`unload()`**.

Si se desean desechar todos de una vez, se puede llamar al método **`clear()`** o **`dispose()`** del `AssetManager`. La diferencia entre ellos es que el primero desecha los assets y remueve todos los assets en cola que no fueron cargados; mientras que el segundo también desecha el `AssetManager`.

## Consultas

La interfaz `Application` provee varios métodos para generar consultas, como por ejemplo obtener el tipo de la aplicación, el consumo de energía, etc.

### Obtener el tipo de aplicación

A veces es necesario saber en qué plataforma está ejecutándose la aplicación. El método `Application.getType()` retorna la plataforma en la que se está ejecutando. En este fragmento de código del juego se puede ver:

```
ApplicationType tipoApp= Application.getType();  
if (tipoApp == ApplicationType.Android) {  
    mapa.update(deltaTime, Gdx.input.getAccelerometerX());  
}
```

## 4.2. *Estudio económico. Valoración de presupuestos*

En primer lugar vamos a mostrar las distintas tarifas que nos vamos a encontrar en el mercado respecto al darse de alta como desarrollador en las plataformas más importantes a día de hoy:

- **Google Play:** El registro como desarrollador Android en Google Play tiene un coste de 25 dólares en un sólo pago, no existe cuota anual.
- **App Store:** El registro en el programa iOS Developer para poder publicar tus propias aplicaciones tiene un coste de 99 dólares anuales.
- **Store Market de Windows:** El registro como desarrollador en esta plataforma tiene un coste de 99 dólares anuales, al igual que la App Store.

Costes de hardware y dispositivos físicos necesarios para la elaboración y fase de pruebas de la aplicación:

- Dos portátiles o equipos de sobremesa con unas características mínimas que garanticen el buen desarrollo de la aplicación: 1600 €.
- Un dispositivo móvil de última generación con sistema Android: 200 €.
- Una Tablet con sistema Android: 270 €.
- (Opcional) Una Tablet o dispositivo móvil con sistema iOS: 600 €.

Por último los distintos costes de software necesario para la realización de la aplicación:

- Adobe Photoshop: 1000 €.

## 4.3. *Elección y justificación de las tecnologías seleccionadas*

Debido a que nuestra aplicación tiene funcionalidades de videojuegos, la elección de las distintas tecnologías que vamos a utilizar va a depender del motor de juegos con el que elaborar la aplicación. Para ello vamos a comentar previamente dos de los motores de videojuegos más usados a día de hoy para este tipo de aplicaciones:

- **AndEngine:** Un motor de videojuegos 2D para Android.

Una de las características que más resalta de AndEngine es que es de código abierto y que en sí, se trata de una implementación 2D de OpenGL para Android por lo que utiliza puramente el lenguaje Java, a diferencia de los motores Unity3D y Shiva3D con los que podríamos trabajar con lenguajes como C#,

Python, o incluso Javascript.

Uno de los puntos en contra de AndEngine es que la documentación "oficial" es aún muy pobre, por lo ni siquiera en la Wiki lograremos encontrar toda la información que necesitamos. Afortunadamente, en la web podemos encontrar algunos links muy interesantes a tutoriales para crear juegos con AndEngine y con ellos podremos iniciarnos con éxito en este motor de juegos.

- **libGDX:** Un motor de videojuegos 2D-3D multiplataforma.

libGDX es un framework para el desarrollo de videojuegos escrito en Java con sus partes más críticas implementadas en C/C++. Corre sobre OpenGL ES 1.0 y 2.0 para dispositivos actuales. Tiene módulos para manejar gráficos, audio y entrada del usuario de una manera muy sencilla que se adapta fácilmente a teclado y ratón como dispositivos táctiles.

El hecho de que sea multiplataforma utilizando java nos ayuda mucho, funciona incluso en iOS, al igual que AndEngine es totalmente libre y gratuito.

**¿Porque libGDX y no AndEngine?** Estos son los puntos principales:

- **Rendimiento:** libGDX es capaz de conseguir una constante 25-30 FPS. En comparación con el 1-2 FPS de AndEngine. En conclusión libGDX escala mucho mejor.
- **La implementación de escritorio:** Gracias a ello se puede probar y desarrollar una aplicación - videojuego en el escritorio. Esto nos permite poder depurar la aplicación con más comodidad sin necesidad de ejecutar un emulador o instalar la aplicación en un dispositivo físico. Y por supuesto, también nos permite desplegar las aplicaciones para Windows, Linux, OS X, etc
- **Comunidad:** La comunidad de libGDX es muy amplia. Los desarrolladores están muy involucrados con la comunidad, las características y correcciones de errores son solucionados diariamente.
- **Ejemplos y Documentación:** Hay una gran cantidad de ejemplos de prueba que podemos descargar, analizar su código y verlo en funcionamiento. Además, el código está totalmente documentado con Javadoc.

La decisión de decantarnos por *libGDX* aparte de los puntos citados anteriormente, fue porque nuestra aplicación debía de elaborarse principalmente tanto para dispositivos Android como para dispositivos iOS. Además la posibilidad de poder compilar también el proyecto para PC, como MAC y HTML era muy tentadora y nos abre un nicho de mercado más amplio.



## Desarrollo de aplicaciones multiplataforma

Esto conlleva que nos adaptemos a ciertas tecnologías que debemos usar al elegir libGDX como base para nuestro proyecto, a continuación una descripción de cada una de ellas:

- **Gradle:** Es una herramienta de gestión de dependencia y para automatizar la construcción de nuestros proyectos, por ejemplo las tareas de compilación, testing, empaquetado y el despliegue de los mismos. Es muy flexible para la configuración, pero además ya tiene armadas las tareas para la mayoría de los proyectos por default. Esta herramienta es usado por grandes proyecto "Open Source" como "Spring", "Hibernate", y "Grails".



- **Eclipse:** Es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. No es más que un entorno de desarrollo integrado (IDE) en el que encontrarás todas las herramientas y funciones necesarias para tu trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

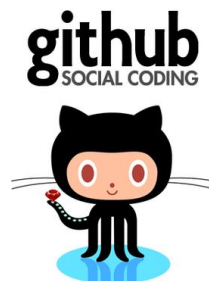




- **Git:** Git es un sistema de control de versiones distribuido cuyo objetivo es el de permitir mantener una gran cantidad de código a una gran cantidad de programadores eficientemente. Git guarda una “foto” (snapshot) del estado de cada archivo en un momento concreto. Si uno de los archivos no ha cambiado no crea una nueva copia del mismo, simplemente crea una referencia al archivo original. Git se basa en que cada programador almacena una copia completa del repositorio en su máquina de forma local, incluido el historial de cambios. Esto implica que muchas de las operaciones realizadas sobre el código fuente no tienen lugar en la red, permitiendo que la velocidad de proceso dependa únicamente en los recursos locales.



- **GitHub:** Un hosting online para nuestros repositorios que utiliza Git para el mantenimiento y versionado del código fuente, añadiendo una serie de servicios extras para la gestión del proyecto y el código fuente.



- **Android:** Es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, inicialmente desarrollado por Android, Inc. Google respaldó económicamente y más tarde compró esta empresa en 2005.



- **Java:** Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser re-compilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web.



- **GWT (Google Web Toolkit):** Es un framework creado por Google que permite ocultar la complejidad de varios aspectos de la tecnología AJAX. Es compatible con varios navegadores, lo cual es notorio ya que cada navegador suele necesitar código específico para lograr un front-end correcto en una aplicación web. El concepto de Google Web Toolkit es bastante sencillo, básicamente lo que se debe hacer es crear el código en Java usando cualquier IDE de Java y el compilador lo traducirá a HTML y JavaScript.



- **LWJGL (Lightweight Java Game Library):** Es una solución dirigida a programadores tanto amateurs como profesionales y está destinada a la creación de juegos de calidad comercial escritos en el lenguaje Java. LWJGL proporciona a los desarrolladores acceso a diversas bibliotecas multiplataforma, como OpenGL (Open Graphics Library) y OpenAL (Open Audio Library), permitiendo la creación de juegos de alta calidad con gráficos y sonido 3D. Por otro lado, LWJGL permite además acceder a controladores de juegos como GamePads, volantes y Joysticks.



- **RoboVM:** Es un plugin que permite desarrollar aplicaciones iOS nativas en Java y poder lanzarlas en el simulador de iOS y dispositivos iOS desde dentro de Eclipse. El tiempo de ejecución es un tiempo similar al de ejecutarla en Android e incluye un puente de Java a Objective-C.



- **Admob:** Es una herramienta que proporciona a los desarrolladores de aplicaciones una solución que les permite distribuir y rentabilizar sus aplicaciones. Todos los SDK se han diseñado para aprovechar las funciones únicas de cada plataforma.

admob ((( )))

- **Texture Packer:** es una herramienta para crear Sprites Sheets. Un Sprite Sheet es una imagen que contiene las animaciones, fondos, objetos, etc. y que va referenciada mediante un .txt o .pack. Los Sprite Sheet se utilizan desde los primeros juegos arcade y ahora funcionan perfectamente con dispositivos móviles. Nosotros lo utilizaremos para generar nuestros TextureAtlas.



Debido a que libGDX requiere imágenes con un tamaño en pixeles que sea potencia de dos y la necesidad de elaborar-retocar imágenes para la interface de la aplicación, se necesita un editor de gráficos, el cual hemos elegido Adobe Photoshop por conocimiento del mismo y por su gran potencial respecto a otras herramientas gratuitas como puede ser Gimp.

- **Adobe Photoshop:** Es un editor de gráficos rasterizados desarrollado por Adobe Systems principalmente usado para el retoque de fotografías y gráficos. Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general.



#### **4.4. Recursos humanos y materiales necesarios para desarrollar el proyecto**

Los recursos humanos necesarios para la elaboración del proyecto:

- Un programador de backend: Juan Maqueda.
- Un programador de frontend: José Iván Jiménez.
- Un diseñador gráfico: Juan Maqueda y José Iván Jiménez con la colaboración de un empleado de la empresa.

Los recursos materiales necesarios para la elaboración y pruebas del proyecto:

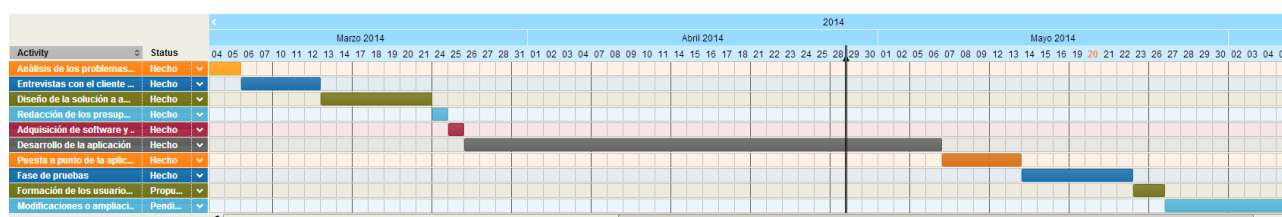
- Tres ordenadores portátiles (2 Intel i5, 8 GB de RAM, 500 GB de HD y un MacBook Air).
- Tablet (Google Nexus 7 y iPad).
- Dispositivos móviles (Sony Experia Z y Samsung Galaxy S2).

## 5. PLANIFICACION TEMPORAL DEL DESARROLLO DEL PROYECTO

### 5.1. Desarrollo del proyecto

En este apartado vamos a tratar la línea de progresión de tiempo de nuestra aplicación, como se llevó a cabo desde el comienzo hasta su puesta en funcionamiento final. Para ello comentaremos paso a paso cada uno de los puntos a tener en cuenta. El tiempo de planificación temporal lo expresaremos en días, contando que un día son alrededor de 8 horas dedicadas.

Diagrama de Gantt del proceso temporal de BabyEduca:



Para una mejor visualización:  
[diagrama\\_gant.PNG](#)

A continuación se explica con más detalle cada paso del proceso:

- **Análisis de los problemas de las aplicaciones actuales.**

Debido a que se quería desarrollar una aplicación para las distintas plataformas móviles, aprovechando el potencial de dicho mercado, se estudió la posibilidad de crear una aplicación que fuese útil, educativa y no estuviese esa idea aun en el mercado. El proceso de búsqueda para saber con más exactitud si esta idea estaba explotada o no, fue de aproximadamente 2 días.

- **Entrevistas con el cliente para obtener las especificaciones del proyecto.**

Ya que la aplicación no va a ser desarrollada para un cliente específico, si no que irá destinada a las plataformas móviles, el número de clientes se amplía exponencialmente. Por lo siguiente se decidió realizar una encuesta a personas conocidas con hijos para saber con mayor exactitud qué requisitos debería de tener la aplicación. Este proceso fue de aproximadamente 5 días.

- **Diseño de la solución a adoptar.**

En base a las encuestas realizadas y la usabilidad que debe tener la aplicación en las tablets se optó por este diseño teniendo en un simple vistazo toda la información, así los niños entenderán mejor el propósito de causa y efecto de la misma. El proceso de diseño fue de unos 7 días.

- **Redacción de los presupuestos.**

La evaluación de los distintos presupuestos que serán necesarios para el desarrollo de la aplicación se realizó durante un día.

- **Adquisición de software y hardware.**

La puesta a punto de los distintos equipos informáticos (instalación en la oficina y configuración de todo el software necesario) se realizó en un día.

- **Desarrollo de la aplicación.**

Este es el proceso más largo de nuestra aplicación, ya que conlleva el estudio previo del motor a usar (libGDX), creación de la estructura base, elaboración y ajuste de la interfaz con dinamismo para cualquier resolución, así como de la elaboración de todas las funcionalidades y su previas investigaciones de uso. El proceso duro 30 días.

- **Puesta a punto de la aplicación.**

Una vez finalizada la versión beta de la aplicación, se tuvo que perfeccionar todo su funcionamiento y hacerla operativa en las distintas plataformas que nos permite libGDX. Este proceso consumió alrededor de 5 días.

- **Fase de pruebas.**

En esta fase se realizan todas las posibles pruebas de la aplicación, posibles bugs, fallos de rendimiento, optimización del uso de los recursos del hardware así como solucionar todo aquello que no funciona como debería. El tiempo dedicado fue de 7 días.

- **Formación de los usuarios finales.**

Debido a que la aplicación es intuitiva y no tenía un cliente directo, no se imparte una formación directa de la misma. Por lo que se realizó un video explicativo y un manual de usuario para facilitar las pequeñas dudas que puedan surgirle al usuario final. El tiempo destinado a este proceso fue de 2 días.

### **5.2. *Lista de modificaciones y ampliaciones futuras que se podrían aplicar a la aplicación proyectada.***

Las modificaciones futuras de BabyEduca van a ser:

1. Rediseño de la interfaz: la usabilidad de la aplicación está un poco limitada, por lo que se tiene en mente hacerle un lavado de cara en versiones posteriores.

2. Implementación de registro de usuarios, pudiendo estos compartir niños, tareas, premios... Por ejemplo si los dos padres tuvieran la aplicación, podría sincronizar esos niños y editarlos cada uno en su dispositivo.
3. Añadir módulo de modificación de niños: ahora mismo la aplicación solo permite el borrado de niños, pero no permite la modificación de sus datos, por lo que se plantea la posibilidad de esta mejora (Desarrollada actualmente).
4. Implementación de versión para Windows Phone: actualmente libGDX no soporta esta plataforma, por lo que sería una buena opción el desarrollo de una versión para esta tecnología, que podría tener un nicho de mercado en un futuro (Actualmente en desarrollo).

### **5.3. Plan de mantenimiento de la aplicación**

Para el mantenimiento de la aplicación en un futuro, se encargará el desarrollador que continúe con la elaboración del proyecto.

Tras elaborar la documentación en Javadoc, tras un previo periodo de estudio de la aplicación, no debería tener problemas en continuar con el desarrollo del proyecto.

Obviamente también cabría la posibilidad de que nosotros continuáramos con el desarrollo, pero ya no como alumnos en prácticas de fct, sino como empleados.

En el apartado de mejoras se pueden observar los puntos más relevantes a tener en cuenta en un futuro, sin tener en cuenta los que se podrían ir añadiendo sobre la marcha.



## 6. ESTUDIO DE VIABILIDAD DEL PROYECTO

### Inversiones

No será necesario el alquiler/compra de ningún local, ya que la aplicación se puede desarrollar en nuestro domicilio. Para el desarrollo si se requiere una inversión en equipos informáticos, dispositivos, software...

- Dos portátiles o equipos de sobremesa con unas características mínimas que garanticen el buen desarrollo de la aplicación: 1600 €.
- Un dispositivo móvil de última generación con sistema Android: 200 €.
- Una Tablet con sistema Android: 270 €.

Por último los distintos costes de software necesario para la realización de la aplicación:

- Adobe Photoshop: 1000 €.

### Financiación

Cada uno de los miembros aporta 1.535 € al desarrollo de la aplicación.

- Capital social (1535 € cada 1) 3.070 €

### Balance

ACTIVO		PASIVO	
<b>A. NO CORRIENTE</b>		<b>FONDOS PROPIOS</b>	
- Equipos informáticos	<b>3.070 €</b>	- Capital Social	<b>3.070 €</b>
<b>Total: 3.070 €</b>		<b>Total: 3.070 €</b>	

### Previsión de ganancias

#### PREVISION DE GASTOS:

- Publicidad: 200 €
- Sueldos y seguridad social: 2.000 €
- TOTAL: 2.500 €

**PREVISION DE INGRESOS EN TRES MESES:**

- Ingresos por ventas (Ventas mensuales \* Coste \* N° Meses) :

$$60 * 0.79 \text{ €} * 3 = 142.2 \text{ €}$$

- Ingresos por publicidad

N° Descargas\* N° Accesos Dia (% Click)\* Dias Totales \* Coste):

$$300 * 0.50 * 90 * 0.10 \text{ €} = 1.350 \text{ €}$$

N° Descargas\* N° Apariciones Publi Dia\* Dias Totales \* Coste):

$$300 * 3 * 90 * 0.01 \text{ €} = 810 \text{ €}$$

- TOTAL: 2.302.2 €

Previsión: + 102,2 € de beneficio.

## 7. DOCUMENTACIÓN DEL DISEÑO E IMPLEMENTACIÓN DE LA SOLUCION ADOPTADA

### 7.1. Prototipo de la aplicación. Diseño de las interfaces

El prototipo de la aplicación y el diseño de las interfaces estaba ya realizado anteriormente, cuando se desarrolló la primera versión de BabyEduca.

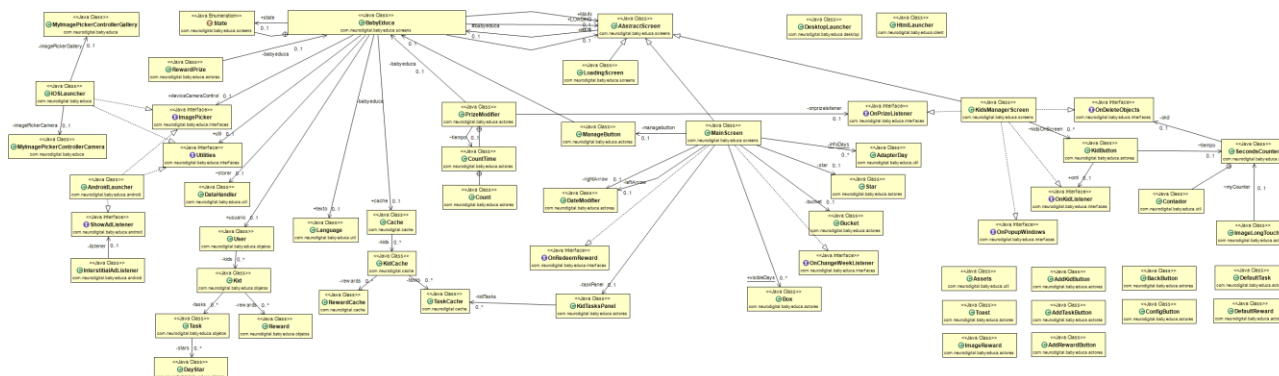
Durante del desarrollo hemos contado con la experiencia del diseñador gráfico de Neurodigital y hemos seguido sus consejos y recomendación para la implementación de mejoras con respecto a la versión anterior.

### 7.2. Diseño lógico de la aplicación a desarrollar, diagramación UML o de Flujo de datos.

A continuación adjuntamos un enlace a la imagen de nuestro diagrama de clases de la aplicación, debido a su tamaño, para una mayor claridad:

[DiagramaClasesBabyEduca.png](#)

Una vista preliminar:



### 7.3. Descripción modular del software a desarrollar

#### 2. Package Screens.

En este paquete encontramos las diferentes pantallas que tendrá nuestra aplicación, en las cuales se visualizara toda la interfaz, se comunicaran nuestros actores y ocurrirá toda la acción de la misma.

- **AbstractScreen:** Pantalla abstracta que podemos extender para crear pantallas nuevas.
- **BabyEduca:** Instancia de la aplicación BabyEduca.

- **KidsManagerScreen:** Ventana de la gestión de los niños tareas y premios.
- **LoadingScreen:** Splash del programa, se mostrara mientras se carguen los recursos.
- **MainScreen:** Clase principal del programa, pantalla principal.

### 3. Package Interfaces.

- **ImagePicker:** Interface para llamar a la cámara y la galería de las respectivas plataformas.
- **OnChangeWeekListener:** Listener relacionado con los cambios de semana.
- **OnDeleteObjects:** Interface relacionada con el borrado de niños, tareas y premios. Mostrar sus respectivos diálogos de confirmación.
- **OnKidListener:** Este listener se usa para cuando se pulsa en un niño, cargara las tareas premios e información relevante de el en pantalla.
- **OnPopupWindows:** Listener usado para lanzar ventanas y diálogos.
- **OnPrizeListener:** Listener implementado en los premios, para que el usuario pueda modificar el coste de cada uno.
- **OnRedeemReward:** Listener implementado en los premios de la ventana principal para poder canjearlos.
- **Utilities:** Interface usada para cosas útiles.

### 4. Package Actores.

- **AddKidButton:** Botón de añadir niño.
- **AddRewardButton:** Botón de añadir premio.
- **AddTaskButton:** Botón de añadir tarea.
- **BackButton:** Botón de volver a ventana principal.
- **Box:** Representa las casillas donde se ponen las estrellas.
- **Bucket:** Cubo de estrellas.

- **ConfigButton:** Botón de configuración de idioma.
- **DateModifier:** Flechas de aumenta o disminuir semanas.
- **DefaultReward:** Premios por defecto.
- **DefaultTask:** Tareas por defecto.
- **ImageLongTouch:** Imágenes con el evento de pulsado largo.
- **ImageReward:** Imágenes de los premios en la ventana principal de la aplicación.
- **KidButton:** Imagen y botón del niño en la pantalla secundario de administración.
- **KidTasksPanel:** Este objeto representa el panel de tareas en la ventana principal.
- **ManageButton:** Botón de la imagen girada del niño, si se pulsa, lanza la ventana de administración.
- **PrizeModifier:** Flechas que modifican el precio de los premios.
- **RewardPrize:** Texto con fondo de estrella con el precio del premio.
- **Star:** Estrella a arrastrar por la pantalla.
- **Toast:** Toast personalizado para libGDX. Muestra notificaciones emergentes.

### 5. Package Objetos.

Aquí volcaremos toda la información que necesitamos guardar, referentes a los niños que se crear sus premios y tareas asignadas, así como las estrellas que se han conseguido.

- **User:** Objeto global del usuario que contiene todos los datos relevantes de su sesión.
- **Kid:** Clase que gestiona el niño.
- **Reward:** Clase que gestiona los premios que tiene un niño.
- **Task:** Clase que gestiona las tareas que tiene un niño.

- **DayStar:** Almacena los días que tienen estrella asignada.

## 6. Package Util.

- **AdapterDay:** Adaptador que genera la posición de los días de la semana, según si es en inglés o español.
- **Assets:** Cadenas de las rutas en los assets de los recursos.
- **DataHandler:** Clase dedicada para extraer los datos del fichero y para guardarlos.
- **Language:** Almacén de variables según idioma.
- **SecondsCounter:** Contador de segundos.

## 7. Package Cache.

- **Cache:** Memoria cache que almacena las imágenes de la aplicación. Sigue la escala jerárquica del objeto User.
- **KidCache:** Cache del niño, que tendrá la lista de las imágenes de tareas y premios, junto con su imagen.
- **TaskCache:** Cache que almacena la imagen de la tarea.
- **RewardCache:** Cache que almacena la imagen del premio.

## 8. Package Android.

- **AndroidLauncher:** Lanzador de la aplicación para Android.
- **InterstitialAdListener:** Listener que gestiona el ciclo de vida del anuncio.
- **ShowAdListener:** Listener que se encarga de mostrar el anuncio cuando este cargado.

## 9. Package iOS.

- **IOSLauncher:** Lanzador de la aplicación para iOS.
- **MyImagePickerControllerCamera:** Controlador de la cámara para iOS.
- **MyImagePickerControllerGallery:** Controlador de la galería para iOS.

## 10. Package Desktop.

- **DesktopLauncher:** Lanzador de la aplicación para escritorio.

### 7.4. *Diseño de la/s bases de datos, con sus diagramas correspondientes.*

Los datos de BabyEduca se almacenan, modifican y cargan mediante objetos.

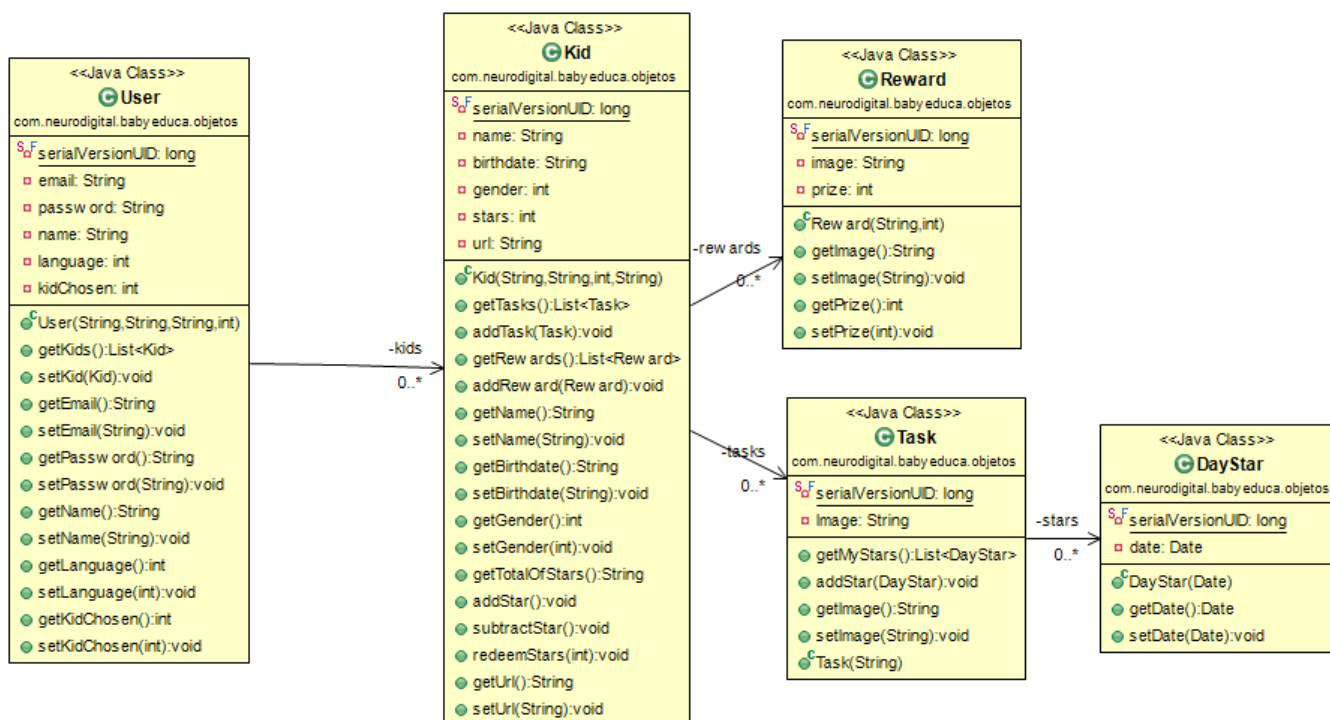
Como se puede observar, el usuario podrá tener 0 o muchos niños, y un niño a su vez podrá tener 0 o muchas tareas y 0 o muchas premios. Con lo consiguiente, una tarea tendrá 0 o muchas estrellas.

Cada clase estará compuesta por sus respectivos constructores y los métodos necesarios para manipular los atributos, obtenerlo, asignarlos y realizar operaciones con ellos.

Cabe destacar que todas las clases disponen de un atributo llamado `serialVersionUID`, que es un atributo generado automáticamente al implementar la interfaz `Serializable`.

Se realiza esto ya que cuando se ejecuta el método `onStop()` en Android, o el `dispose()` en la mayoría de las plataformas soportadas por `libGDX`, estos datos serán volcados en un fichero. Al iniciar la aplicación se leerá el objeto `User` almacenado de la sesión anterior y se recuperaran los datos de esa sesión para volver a trabajar con ellos.

De las imágenes, se almacenará el `path` de cada una para volver a cargarlas posteriormente. De esto se hablará en puntos posteriores.



## 7.5. Otras estructuras de datos que se utilizan en la aplicación.

BabyEduca dispone de otra estructura de datos a la que le hemos llamado caché. Como su nombre indica, hará funciones de cache, en la que se almacenarán las imágenes personalizadas del usuario (fotos de perfil, de tareas creadas...).

Se usa este método porque cargar Textures en tiempo de ejecución requiere un precio demasiado elevado y da la sensación de lentitud.

En esta caché se almacenarán única y exclusivamente las imágenes personalizadas de los niños, tareas y premios a través de la url del path almacenados en la estructura de datos principal.

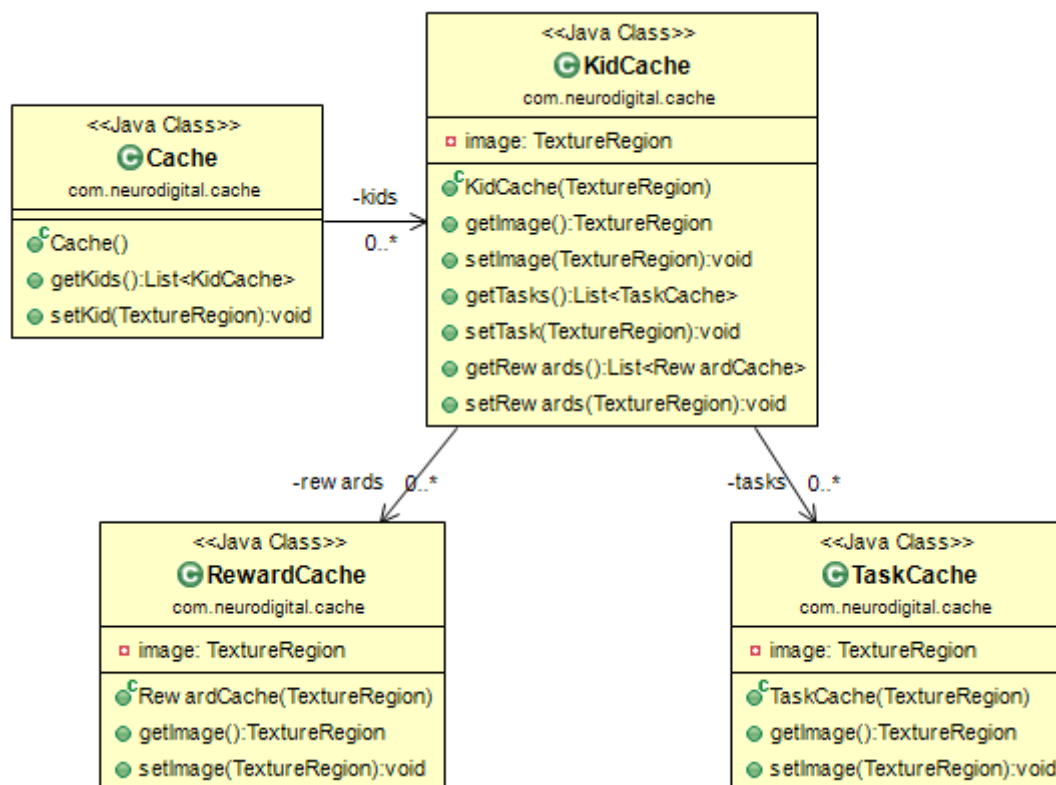
Se cargarán al inicio de la aplicación y el programa las irá usando conforme se vayan necesitando.

Como se puede comprobar la cache constará de:

1. **Cache:** este objeto alojará la lista de niños que dispone el usuario, dando la posibilidad de introducir nuevos en tiempo de ejecución.
2. **KidCache:** es el cache de los niños. Se almacenará el TextureRegion correspondiente del niño, siempre y cuando sea imagen personalizada, la lista de tareas y la lista de premios.
3. **TaskCache:** almacenará el TextureRegion personalizado de la tarea del niño.



4. **RewardCache:** almacenará el TextureRegion personalizado del premio del niño.



## 7.6. Estudio de la seguridad de la aplicación

Existe una gran preocupación referente al tema de seguridad de una aplicación, podríamos diferenciar entre dos posturas distintas:

- Seguridad referente a datos, tanto de usuarios como de servidores:

Nuestra aplicación no tiene ningún problema de seguridad referente a los datos, ya que es una aplicación totalmente local, sin uso de internet, ni datos personales.

- Seguridad referente a que puedan descompilar nuestra aplicación y obtener el código fuente:

Los desarrolladores siempre tendremos la preocupación de que puedan obtener nuestro código fuente con programas al estilo de: DJ Java Decompiler. Para evitar que nuestro código sea desensamblado disponemos de un software ofuscador, como por ejemplo: Mocha Source Obfuscator.

## 8. CODIGO FUENTE DOCUMENTADO

Cabe destacar que libGDX no da soporte a elementos nativos de las diferentes plataformas, por lo que para su uso, es necesario la implementación de estos elementos según en qué plataforma se esté ejecutando.

Para la solución de este problema se ha implementado una interface que gestionará la cámara y la galería con sus métodos sobrescritos en cada proyecto.

La interface es la siguiente:

```
/**
 * Interface para llamar a la camara y la galeria de las respectivas
 * plataformas
 * @author Juan, Ivan
 */
public interface ImagePicker {
    /**
     * Lanza la camara, permitiendonos sacar imagenes
     * @return true o false
     */
    boolean startPreview();
    /**
     * Devuelve los bytes de la imagen para trabajar con ellos
     * @return
     */
    byte[] getBytes();
    /**
     * Cuando se acaba de guardar la imagen, salta el evento y eliminara los bytes de la imagen
     * de la memoria
     */
    void pictureJustSaved();
    /** Abre la galeria para seleccionar una imagen*/
    void openGallery();
}
```

Cada proyecto implementa un constructor del proyecto core de libGDX. Así que en el constructor del core necesitará la referencia a la interface implementa en dichos proyectos.

De esta manera si se necesita especificar elementos nativos de cada plataforma solo bastará con genera una interface e implementar los métodos que necesitemos. La interface será nuestro "puente" de comunicación entre plataformas.

En el constructor del proyecto core habrá que pasarle como referencia la interface implementada en cada proyecto específico, para que pueda sobrescribir los métodos.

```
public BabyEduca(ImagePicker imagePicker, Utilities utility) {
    this.imagePicker = imagePicker;
    this.util = utility;
}
```

## 2. Implementación de Cámara

- Implementación de la cámara en Android

La actividad principal del proyecto de Android deberá implementar esta interface e implementar el código nativo para cada acción que vamos a necesitar.

En el objeto de llamada al core, le pasará una referencia local mediante **this**.

```
// Se crea el view de babyeduca
View gameView = initializeForView(new BabyEduca(this, this), cfg);
```

Hecho esto, ya el siguiente paso es codificar el código nativo en cada override.

### Boolean startPreview()

```
@Override
public boolean startPreview() {
    //Fichero de la imagen a crear
    File f = null;
    //Se crean el intent que llamara a la camara
    i = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);

    try {
        //Prepara el fichero, recoge el path y se añade
        //el uri del fichero como extra
        f = setUpPhotoFile();
        mCurrentPhotoPath = f.getAbsolutePath();
        i.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
    } catch (IOException e) {

        e.printStackTrace();
        f = null;
        mCurrentPhotoPath = null;
    }

    startActivityForResult(i, RESULT_CAMERA_IMAGE);
    return true;
}
```

Para llamar a la cámara por defecto de Android, se puede usar un Intent, y no tener que implementar una cámara personalizada desde 0.

El proceso de esta acción es:

- Generar el intent de la cámara
- Generar el fichero en el que guardaremos la imagen tomada.
- Obtener el path del fichero
- Añadirle el fichero como extra al intent
- Lanzar el intent mediante startActivityForResult

Para generar el fichero se ha elaborado un metodo que nos facilita el trabajo.

### File setUpPhotoFile()

```
/**
 * Prepara el fichero para la imagen, y obtiene el path absoluto
 *
 * @return file
 * @throws IOException
 */
private File setUpPhotoFile() throws IOException {
    File f = createImageFile();
    mCurrentPhotoPath = f.getAbsolutePath();

    return f;
}
```

Dentro de este método se realizarán las acciones de crear el fichero y obtener el path del fichero.

Para crear le fichero, se llama a otro método que lo generará-

### File createImageFile()

```
/**
 * Crea un file para guardar la imagen
 *
 * @return a file for the image
 * @throws IOException
 */
private File createImageFile() throws IOException {
    //Crea el nombre de la imagen
    String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss",
        Locale.FRANCE).format(new Date());
    String imageFileName = "JPEG_" + timeStamp + "_";
    //Directorio donde almacenara la imagen al echarla
    File storageDir = Environment
        .getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
    //Crea el archivo temporal de la imagen
    File image = File.createTempFile(imageFileName, /* prefix */
        ".jpg", /* suffix */
        storageDir /* directory */
    );

    //Path del fichero
    mCurrentPhotoPath = "file:" + image.getAbsolutePath();

    return image;
}
```

Dentro de este método se capturará el momento en el que se creo el fichero a partir de un objeto Date.

El nombre del fichero lo generará con la cabezerá JPEG y la fecha que se guardó.

Posteriormente se obtiene el directorio donde guarda las imágenes Android de las aplicaciones

Teniendo ya el nombre y el directorio, el siguiente paso es crear el archivo temporal mediante el método que nos ofrece la clase File `createTempFile`, pasándole como parametros el nombre del fichero, la extensión y el directorio donde lo creará.

El siguiente paso en el proceso será recuperar los datos recibidos en el método **`startActivityResult()`**

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    // Caso Camara
    //Se configuran las opciones del bitmap para
    //reducir la memoria que ocupara en ram
    BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = false;
    options.inPreferredConfig = Config.RGB_565;
    options.inDither = true;
    if (resultCode == Activity.RESULT_OK
        && requestCode == RESULT_CAMERA_IMAGE) {
        //Añade la foto a la galeria
        galleryAddPic();
        //Se decodifica el archivo en un bitmap
        bitmap_original = BitmapFactory.decodeFile(mCurrentPhotoPath,
            options);
        //Se comprime el bitmap en jpeg y al 50% de calidad
        //en el outputStream
        ByteArrayOutputStream stream = new ByteArrayOutputStream();
        bitmap_original.compress(Bitmap.CompressFormat.JPEG, 50, stream);
        bitmap_original.recycle();
        bitmap_original = null;
        //Obtiene los bytes, que se usaran en libgdx
        byteArray = stream.toByteArray();
        getBytes();
    }
}
```

Los pasos que realiza en el metodo resumidamente son:

- Crear las opciones de bitmap para reducir memoria.
- Si el usuario acepta la imagen, se añade a la galería
- Se obtiene el bitmap a partir del path de la imagen.
- Se crea un flujo para obtener los array de bytes, que es lo que se le pasará a libGDX.

El método de añadir la imagen a la galería es util ya que siempre podremos tener esa imagen accesible en la galería de nuestro teléfono, para otros usos.

### galleryAddPic()

```
/**
 * Añade la foto a la galeria a traves de broadcast
 */
private void galleryAddPic() {
    Intent mediaScanIntent = new Intent(
        "android.intent.action.MEDIA_SCANNER_SCAN_FILE");
    File f = new File(mCurrentPhotoPath);
    Uri contentUri = Uri.fromFile(f);
    mediaScanIntent.setData(contentUri);
    this.sendBroadcast(mediaScanIntent);
}
```

Para que Android detecte esta imagen en la galería se le tiene que enviar el fichero de la imagen mediante un BroadcastReceiver.

Para hacer esto se genera un intent con la acción a realizar, y se le pasa como datos un uri con el fichero generado.

El método `getBytes()`, está implementado en el render de libGDX, esto quiere decir que esta en un bucle repetitivo que nunca termina, y estara comprobando que los bytes no sean null. Cuando el objeto se carga con los bytes de la imagen, libGDX lo detectará y procesará esa imagen.

```
@Override
public byte[] getBytes() {
    return byteArray;
}
```

Asi que para que no procesara la imagen indefinidas veces, el método `pictureJustSaved()` se lanzará cuando la imagen ha sido ya procesada por libGDX, poniendo el `byteArray` a null, con lo que ya no entrará mas en el render.

```
@Override
public void pictureJustSaved() {
    byteArray = null;
}
```

- **Implementación de la cámara en iOS**

El desarrollo de elementos nativos para iOS se hace mediante RoboVM, una librería que nos permite programar en Java y ella importa el código fuente a nativo de iOS.

En iOS, la cámara y galería pertenecen al grupo UIImagePickerController, por lo que se ha creado una clase para cada una y personalizarlas.

### **MyImagePickerControllerCamera**

```
import org.robvm.apple.uikit.UIImagePickerController;

/**
 * UIImagePickerController de iOS personalizado a que no rote y este en portrait
 * @author Neurodigital Technologies
 */
public class MyImagePickerControllerCamera extends UIImagePickerController {
    @Override
    public boolean shouldAutorotate() {
        return false;
    }

    @Override
    public UIInterfaceOrientationMask getSupportedInterfaceOrientations() {
        return UIInterfaceOrientationMask.All;
    }

    @Override
    public UIInterfaceOrientation getPreferredInterfaceOrientation() {
        return UIInterfaceOrientation.Portrait;
    }
}
```

Se han sobrescrito los métodos que permiten la rotación, las orientaciones soportadas, la orientación asignada y la ocultación del status bar de iOS.

En los métodos sobrescritos de nuestra interface UIImagePickerController tendrán el siguiente código:

## Boolean startPreview()

```
@Override
public boolean startPreview() {
    UIWindow = UIApplication.getSharedApplication().getKeyWindow();
    //Se crea el imagePicker de la cámara, permitiendo editar la imagen capturada
    imagePickerCamera = new MyImagePickerControllerCamera();
    imagePickerCamera.setAllowsEditing(true);
    imagePickerCamera
        .setModalPresentationStyle(UIModalPresentationStyle.CurrentContext);
    imagePickerCamera
        .setSourceType(UIImagePickerControllerSourceType.Camera);
    //Se añade el imagepicker al rootView principal
    rootVC = UIWindow.getRootViewController();
    rootVC.presentViewController(imagePickerCamera, true, null);
    //Listener del imagePicker que devuelve la imagen tomada
    UIImagePickerControllerDelegateAdapter delegate = new UIImagePickerControllerDelegateAdapter() {
        @Override
        public void didFinishPickingImage(UIImagePickerController picker,
            UIImage image, NSDictionary info) {
            System.out.println(image.getOrientation());
            //Se obtienen los bytes de la imagen
            NSData bytes = image.toPNGData();
            byteArray = bytes.getBytes();
            //Quita el imagepicker de la pantalla
            picker.dismissViewController(true, null);
        }
    };
    //Se añade el listener
    imagePickerCamera.setDelegate(delegate);
    imagePickerCamera.addStrongRef((ObjCObject) delegate);
    return true;
}
```

El proceso es bastante simple. Los pasos son los siguientes:

1. Se crea el objeto de la cámara.
2. Se le añade el estilo y algunas opciones (como por ejemplo, que permita editar la imagen sacada).
3. Se añade la cámara al rootView principal de la aplicación, que hará que entre en primer plano y se pueda usar.
4. Se crea el listener que devolverá la imagen y gestionará el proceso de codificación de los bytes, como en las otras plataformas.
5. Se le añade el listener al objeto.

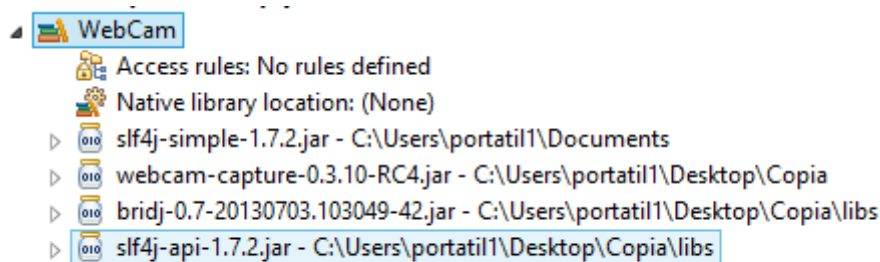
Cuando se obtienen los bytes, libGDX los gestionará en el proyecto core. Los métodos sobrescritos **pictureJustSaved()** y **getBytes()** son exactamente igual que en Android, por lo que no hace faltacomentarlos.

### • Implementación de la cámara en Desktop

Para la implementación de la cámara en Desktop, se ha hecho uso de unas librerías externas elaboradas en Java, que facilitan el desarrollo e implementación de la web cam. En la bibliografía se puede consultar los respectivos enlaces.



Los jar exportados son:



Hecho esto, ya queda sobrecribir los métodos de la interface ImagePicker.

Ya que el método startPreview es bastante extenso, se irá explicando paso a paso.

```
// Se obtiene la camara por defecto
try {
    webcam = Webcam.getDefault();
} catch (WebcamException wb) {

    return false;
}
```

Para obtener la previsualización de la web cam, mediante la clase Webcam hay varios métodos con los que nos facilita el uso de esta.

El método getDefault() nos devuelve el objeto de la cámara detectada en el equipo por defecto. Si no tuviera ninguna, saltaría una excepción, así que saldríamos del startPreview devolviendo false.

```
// Se le añade la maxima resolucio soportada
Dimension di = new Dimension(
    webcam.getViewSizes()[webcam.getViewSizes().length - 1].width,
    webcam.getViewSizes()[webcam.getViewSizes().length - 1].height);
webcam.setViewSize(di);
```

El siguiente paso es asignarle las dimensiones de la previsualización, que será el máximo ancho y alto soportado.

```
// Se crea la previsualizacion de la camara
WebcamPanel panel = new WebcamPanel(webcam);
panel.setFPSDisplayed(true);
```

El objeto WebcamPanel, será el panel en el cual mostrará la previsualización.

Una de las opciones que permite es mostrar los fraps por segundo de la cámara, algo que nos pareció interesante.

```
// Ventana en la que se mostrara la camara
final JFrame window = new JFrame("Webcam Monitor");
// Boton de sacar foto
JButton button = new JButton("Hacer foto");
button.setAlignmentX(Component.CENTER_ALIGNMENT);
button.addActionListener(new ActionListener() {

    @Override
    public void actionPerformed(ActionEvent e) {
        // Se hace una imagen y se cierra la camara
        BufferedImage image = webcam.getImage();
        webcam.close();

        // Se obtiene el bytearray de la imagen
        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        try {
            ImageIO.write(image, "png", baos);
            baos.flush();
            imageInByte = baos.toByteArray();
            baos.close();
            window.dispose();

        } catch (IOException e1) {

            e1.printStackTrace();

        }

    }

});
```

Para que el usuario pueda interactuar, se ha usado elementos de Swing para añadir un botón de tomar foto.

Lo primero es crear el frame donde se añadirán los dos elementos.

Acto seguido se crea el botón, que irá alineado en el centro del frame.

Se le añade su listener, en el cual se le implementará un buffer de imágenes. Con el buffer ya conseguido, se crea un flujo de salida que nos permitirá escribir la imagen en bytes y libGDX pueda trabajar con esos bytes.

Por último, en el listener, se cierran los recursos usados, así como la previsualización, si todo fue correcto.

```
// Se crea el layout del jframe, el tamaño, y se le añaden
// los elementos
window.setLayout(new BorderLayout(window.getContentPane(),
    BorderLayout.Y_AXIS));
window.setSize(600, 600);
window.add(panel);
window.add(button);
window.setResizable(true);
window.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
// Cuando se cierre la ventana, se apaga la web cam
window.addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        super.windowClosing(e);
        webcam.close();
    }
});

window.pack();
window.setVisible(true);
return true;
```

Hecho el botón, lo siguiente es parametrizar el frame.

Se le añade un layout que alinea verticalmente, parecido al linearLayout de Android y varios parámetros más.

Se le ha añadido un listener que detecta cuando la ventana se cierra, para que no se quede el proceso en memoria cargado de la web cam.

Una vez listo, se comprimen los elementos y se muestra en el monitor.

Los demás métodos son exactamente igual que en el ejemplo de Android, así que no hay necesidad de comentarlos de nuevo.

### 3. Implementación de Galería

- Implementación de la galería en Android

Para implementar el evento de la galería, hay que sobrescribir el método `openGallery()` en las plataformas que deseemos que tenga esta función.

Veamos el código:

```
@Override
public void openGallery() {
    //Crea intent para abrir la galeria
    i = new Intent(Intent.ACTION_PICK,
        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
    startActivityForResult(i, RESULT_LOAD_IMAGE);
}
```

Para llamar a la galería de Android es bastante simple. Se crea un intent con la acción a realizar y pasándole el uri del ContentProvider que usa.

Se lanza el `startActivityResult` y se esperan resultados.

```
} else if (resultCode == Activity.RESULT_OK
    && requestCode == RESULT_LOAD_IMAGE) {
    //Se obtiene el uri con la imagen seleccionada
    Uri selectedImage = data.getData();
    //Datos con las iamgenes de la galeria
    String[] filePathColumn = { MediaStore.Images.Media.DATA };
    //Consulta para obtener la imagen seleccionada
    Cursor cursor = getContentResolver().query(selectedImage,
        filePathColumn, null, null, null);
    cursor.moveToFirst();

    //Se obtiene la columna de la imagen seleccionada
    int columnIndex = cursor.getColumnIndex(filePathColumn[0]);
    //Se obtiene el path de la imagen
    String picturePath = cursor.getString(columnIndex);
    cursor.close();
    //Se obtiene el bitmap de la imagen
    bitmap_original = BitmapFactory.decodeFile(picturePath, options);
    //Se recogen los bytes para usarlos en libgdx
    ByteArrayOutputStream stream = new ByteArrayOutputStream();
    bitmap_original.compress(Bitmap.CompressFormat.JPEG, 50, stream);
    bitmap_original.recycle();
    bitmap_original = null;
    byteArray = stream.toByteArray();
    getBytes();
}
```

Si el resultado es positivo, se obtiene el uri de la imagen seleccionada y se hace una consulta de esa imagen al almacenamiento de Android, que nos devolverá el path de la imagen.

A partir del path se obtiene el bitmap, y a través del flujo de bytes se obtienen los bytes del bitmap, los cuales usará libGDX.

Despues de procesar la imagen de la galería, los bytes se ponen a null con el método `pictureJustSaved()` y termina el proceso.

### • Implementación de la galería en iOS

La implementación de la galería es muy parecida a la implementación de la cámara en iOS. Como la galería también es un `UINavigationController`, se crea una clase que extienda de esa clase y se personaliza.

```
import org.robovm.apple.uikit.UIImagePickerController;

/**
 * UIImagePickerController de la galeria de iOS personalizado a que rote, y se abra en landscape.
 * Tambien oculta el status bar
 * @author Neurodigital Technologies
 */
public class MyImagePickerControllerGallery extends UIImagePickerController {
    @Override
    public boolean shouldAutorotate() {
        return true;
    }

    @Override
    public UIInterfaceOrientationMask getSupportedInterfaceOrientations() {
        return UIInterfaceOrientationMask.All;
    }

    @Override
    public UIInterfaceOrientation getPreferredInterfaceOrientation() {
        return UIInterfaceOrientation.LandscapeLeft;
    }

    @Override
    public boolean prefersStatusBarHidden() {
        return true;
    }
}
```

Hecho esto, hora de sobrescribir el método openGallery()

## OpenGallery()

```
@Override
public void openGallery() {

    UIWindow = UIApplication.getSharedApplication().getKeyWindow();
    //Se crea el imagepicker de la galeria, permitiendo editar la imagen seleccionada
    imagePickerGallery = new MyImagePickerControllerGallery();
    imagePickerGallery.setAllowsEditing(true);
    imagePickerGallery
        .setModalPresentationStyle(UIModalPresentationStyle.CurrentContext);
    imagePickerGallery
        .setSourceType(UIImagePickerControllerSourceType.PhotoLibrary);
    //Se añade al rootView principal el imagepicker de la galeria
    rootVC = UIWindow.getRootViewController();
    rootVC.presentViewController(imagePickerGallery, true, null);
    //Se crea el listener de la galeria, que nos permitira trabajar con la imagen seleccionada
    UIImagePickerControllerDelegateAdapter delegate = new UIImagePickerControllerDelegateAdapter() {
        @Override
        public void didFinishPickingImage(UIImagePickerController picker,
            UIImage image, NSDictionary info) {
            System.out.println(image.getOrientation());
            //Se pasa a bytes la imagen seleccionada
            NSData bytes = image.toPNGData();
            byte[] byteArray = bytes.getBytes();
            //Se quita el imagepicker de la pantalla
            picker.dismissViewController(true, null);
            UIApplication.getSharedApplication().setStatusBarHidden(true);
        }
    };
    //Se le añade el listener
    imagePickerGallery.setDelegate(delegate);
    imagePickerGallery.addStrongRef((ObjCObject) delegate);
}
```

El proceso es casi identico al de la cámara. Lo unico que se modifica es el método setSourceType, en el que se le pasa como parámetro que sea la librería de fotos de iOS.

El listener y demas creado hace la misma funcionalidad.

Después de obtener la imagen y procesarla libGDX mediante el método `getBytes`, se sobrescribe el método `pictureJustSaved()` como en el ejemplo de Android.

### • Implementación de la galería en Desktop

Sobreescribimos el método `openGallery()`, igual que en las demás plataformas. Vayamos por partes:

Galería en si no hay en desktop, asique se ha decido usar un `JFileChooser` de Swing para implementar la selección de imágenes.

El primer paso antes de la implementación del `JFileChooser` es la creación de un frame.

```
// Frame invisible, permitiendo que aparezca la ventana en  
// primer plano  
Frame frame = new Frame();  
frame.setUndecorated(true);  
frame.setOpacity(0);  
frame.setLocationRelativeTo(null);  
frame.setVisible(true);  
frame.toFront();  
frame.setVisible(false);  
frame.dispose();
```

El frame sera invisible, sin decorado, y sin localización aparente. Lo único relevantes es que estará colocado en primer plano. Se hace esto ya que `JFileChooser`, si se lanza sin ningun padre relativo en el constructor, aparecerá detrás de la ventana que este en primer plano.

Esto era un problema porque siempre aparecia detrás de la aplicación de BabyEduca. Con esto aparecerá en segundo plano detrás del frame invisible, por lo que visiblemente al usuario sera como si estuviera en primer plano.

```
// Se crea un filechooser de imagenes  
JFileChooser fc = new JFileChooser();  
fc.setFileSelectionMode(JFileChooser.FILES_ONLY);  
fc.setFileFilter(new FileNameExtensionFilter("Images", "bmp",  
"jpg", "png"));
```

Hecho esto, se construlle el `JFileChooser`, añadiendole como filtros que solo se puedan seleccionar ficheros, y no directorios, y de filtro que solo se puedan seleccionar imágenes con extensión jpg, png y bmp.

```
int answer = fc.showOpenDialog(null);
// Si se elige una imagen
if (answer == JFileChooser.APPROVE_OPTION) {
    // Apuntador al fichero elegido
    File fileSelected = fc.getSelectedFile();
    FileInputStream fis = null;
    try {
        fis = new FileInputStream(fileSelected);
    } catch (FileNotFoundException e) {

        e.printStackTrace();
    }

    ByteArrayOutputStream bos = new ByteArrayOutputStream();
    byte[] buf = new byte[1024];
    try {
        // Se leen los bytes de la imagen
        for (int readNum; (readNum = fis.read(buf)) != -1;) {
            // y los escribe en el output
            bos.write(buf, 0, readNum);
        }
    } catch (IOException ex) {

    }

    // Se guardan los bytes en un array, pasandolos a libgdx
    imageInByte = bos.toByteArray();
}
```

Por último, se muestra el JFileChooser, y guardamos en una variable entera el resultado de la operación. Si el resultado es igual al de selección de un fichero, se procesa, y a través de flujos se obtienen los bytes de la imagen, que los procesará libGDX en la aplicación.

Posteriormente, con el método pictureJustSaved() se pondrá el byteArray a null y concluirá la operación.

## 4. Implementación de Toast personalizado

Debido a que requeríamos del uso de un Toast, inicialmente se usó para la versión de Android un Toast nativo. Pero más adelante necesitaríamos para el resto de las plataformas, dar una notificación emergente. Con lo cual, ya que con libGDX podemos realizar animaciones, se decidió realizar nuestro propio Toast personalizado usarlo en todas las plataformas por igual.

En primer lugar creamos una clase Toast que extiende de Actor creándole las variables necesarias.



```
package com.neurodigital.babyeduca.actores;

import com.badlogic.gdx.Gdx;

/**
 * Toast personalizado para libGDX. Muestra notificaciones emergentes.
 *
 * @author Juan, Ivan
 */
public class Toast extends Actor {
    private String message;
    private TextureRegion background;
    private BitmapFont font;
    private float desp;
    private Stage stage;
```

Seguidamente creamos el constructor de la clase, al cual será necesario pasarle ciertos parámetros para la creación del mismo, ya que nuestro Toast es personalizado, está recogida la posibilidad de indicarle la imagen de fondo y tipo de fuente que se desee, para así hacer realmente una mayor personalización y poder reutilizar la clase en aplicaciones futuras con otro diseño.

```
/**
 * Toast personalizado para libGDX. Muestra notificaciones emergentes.
 *
 * @param message
 *         Texto que se mostrara en el Toast
 * @param background
 *         Imagen del toast
 * @param font
 *         Tamaño del texto a mostrar
 * @param stage
 *         Stage al que se añade el actor toast
 */
public Toast(String message, TextureRegion background, BitmapFont font, Stage stage) {
    /** Accion para remover el toast */
    RemoveActorAction remov;
    /** Acciones de movimiento del toast */
    MoveByAction movein, moveout, stop;
    this.message = message;
    this.font = font;
    this.background = background;
    this.desp = this.font.getBounds(this.message).width;
    this.stage = stage;
```

A continuación, le indicamos el tamaño que tendrá nuestro actor en escena y la posición inicial del mismo.



```
/** Tamaño dinamico del toast segun longitud(desp) del texto ha mostrar */
setSize(desp, (Gdx.graphics.getWidth() * 0.20f) / 5.88f);
/**
 * Posicion dinamica del toast segun resolucio y longitud(desp) del
 * texto ha mostrar
 */
setPosition(
    (Gdx.graphics.getWidth() * 0.50f - (desp / 2) - (desp * 0.15f)),
    0);
```

Debido a que deseamos realizar la simulación de una notificación emergente, le hemos añadido a nuestro actor una serie de animaciones para crear dicho efecto.

```
/** Accion remover toast */
remov = new RemoveActorAction();
remov.setRemoveActor(this);

/** Accion FadeIn toast */
movein = new MoveByAction();
movein.setAmount(0, Gdx.graphics.getHeight() * 0.15f);
movein.setDuration(0.5f);
movein.setInterpolation(Interpolation.swingOut);

/** Accion detener toast */
stop = new MoveByAction();
stop.setAmount(0, 0);
stop.setDuration(1f);

/** Accion FadeOut toast */
moveout = new MoveByAction();
moveout.setAmount(0, -Gdx.graphics.getHeight() * 0.15f);
moveout.setDuration(0.2f);

/** Añadimos la distintas acciones mediante una secuencia */
this.addAction(Actions.sequence(movein, stop, moveout, remov));
```

Dibujamos nuestro Toast en pantalla, tanto el fondo como el texto indicado.

```
/**
 * Dibujamos el actor (mensaje e imagen)
 */
@Override
public void draw(Batch batch, float parentAlpha) {
    batch.draw(background, getX(), getY(), getOriginX(), getOriginY(),
        getWidth() + (getWidth() * 0.30f), getHeight(), getScaleX(),
        getScaleY(), getRotation());
    font.draw(batch, message, getX() + (getWidth() * 0.15f), getY()
        + (getHeight() * 0.80f));
}
```

Y lo cargamos en pantalla (Stage establecido al crearlo).

```
/**
 * Muestra el Toast en pantalla
 */
public void show() {
    stage.addActor(this);
}
```

Finalmente hemos creado un método para optimizar código, el cual genera nuestro Toast según los parámetros indicados.

```
/**
 * Crea un nuevo Toast
 * @param message Mensaje del toast
 * @param background Fondo del toast
 * @param font Fuente del toast
 * @param stage Stage al que se añade el actor toast
 * @return Objeto toast listo para usar
 */
public static Toast makeText(String message, TextureRegion background, BitmapFont font, Stage stage) {
    return new Toast(message, background, font, stage);
}
```

Veamos un ejemplo de llamada a nuestro Toast:

```
Toast.makeText(babyeduca.texto.Toast_NewTaskReward_picture,
    toast_background, babyeduca.fuenteBase, stage)
    .show();
```

## 5. Interfaz dinámica

El desarrollo de todo el Frontend de la aplicación ha sido enfocado para que se genere dinámicamente, independientemente del dispositivo móvil, este dinamismo se genera según la resolución de pantalla del dispositivo.

Debido a que la proporción de las resoluciones de pantalla de los distintos dispositivos móviles son diferentes, se tuvo que generar un método que calculase según el ancho de la imagen la proporción de altura de la misma según la resolución de la pantalla donde se ejecuta la aplicación:

```
/**
 * Calculamos el porcentaje de escala del alto de la imagen segun un ancho
 * indicado.
 *
 * @param widthPercent
 * @return
 */
public float imageScale(float widthPercent) {
    float alto, por cientoAlto = 0;
    float ancho = anchura;
    alto = altura;
    por cientoAlto = (ancho * widthPercent) / alto;
    return por cientoAlto;
}
```

Veamos un ejemplo de uso:

```
btn_task_camera = setImageButton(camera_icon,
    windowAddTasks.getWidth() * 0.61f,
    windowAddTasks.getHeight() * 0.23f,
    windowAddTasks.getWidth() * 0.08f, windowAddTasks.getHeight()
    * babyeduca.imageScale(0.08f));
```

Ademas de la situación anterior, ciertas imágenes no deben de perder proporción y a su vez deben verse bien en todos los dispositivos y centradas en cada pantalla según sus dimensiones. Para ello usamos el siguiente código:

```
babyeduca.SB.draw(splashLogo, babyeduca.anchura * 0.50f - (babyeduca.anchura*0.50f / 2),
    babyeduca.altura * 0.50f - (((babyeduca.anchura*0.50f)/2.169f) / 2),babyeduca.anchura*0.50f,
    ((babyeduca.anchura*0.50f)/2.169f));
```

Para centrar la imagen debemos posicionarla en el centro del alto y el ancho de la pantalla y restarle la mitad de su tamaño respectivamente.

## 6. Admob

En la aplicación de Android, se ha incluido publicidad para la versión gratuita. La tecnología usada para ello es Admob, una librería elaborada por Google que nos ofrece la posibilidad de usar banners, anuncios de pantalla completa, y unos cuantos más.

Nosotros nos decantamos por el anuncio de pantalla completa, que se mostrará cuando surgen unos determinados eventos en la aplicación.

Para poder usar Admob, antes que nada, debemos tener instalados los google play services y configurar el proyecto para el uso de la librería. Se puede seguir los pasos para ello en <http://developer.android.com/google/play-services/setup.html>.

Una vez hecho este paso, hay que añadir una serie de permisos al Manifest, así como una activity especializada en el anuncio.  
Permisos de Internet:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Metadatos de google play services:

```
<meta-data android:name="com.google.android.gms.version"  
    android:value="@integer/google_play_services_version"/>
```

Declarar AdActivity:

```
<activity android:name="com.google.android.gms.ads.AdActivity"  
    android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize"/>
```

Una vez hecho esto, se declaran dos variables que se usaran en el proceso, la variable del anuncio y un String con el código que te proporciona Google al registrarte.

```
private PublisherInterstitialAd interstitial;  
private static final String AD_UNIT_ID_INTERSTITIAL = "ca-app-pub-";
```

Para construir el anuncio, se le añade el código y un listener que se usará para mostrar el anuncio.

```
// se crea el interstitial anuncio  
interstitial = new PublisherInterstitialAd(this);  
interstitial.setAdUnitId(AD_UNIT_ID_INTERSTITIAL);  
interstitial.setAdListener(new InterstitialAdListener(this));
```

Para mostrar el anuncio desde Libgdx se ha usado una interface que sobreescribe el método **showAd()**. Este método recoge como parámetros dos

booleanos, uno para controlar que lo muestre o no, y otro para controlar que sea en modo test o no.

```
/**
 * Interface usada para cosas utiles
 *
 * @author Juan, Ivan
 *
 */
public interface Utilities {

    /**
     * Lanza el ad de publicidad
     */
    void showAd(boolean show, boolean testMode);
}
```

Y el método sobrescrito quedaría de la siguiente manera:

```
@Override
public void showAd(boolean show, boolean testMode) {
    if(show) {
        loadInterstitial(testMode);
    }
}
```

Dentro de este método, se llama otro método que se encarga de cargar el anuncio. Veámoslo:

```
/** Cuando se carga lo muestra */
public void loadInterstitial(final boolean testMode) {
    uiThread.post(new Runnable() {
        public void run() {
            //Se construye el publicador de anuncio
            PublisherAdRequest adRequest;
            if(testMode){
                adRequest = new PublisherAdRequest.Builder()
                    .addTestDevice(tm.getDeviceId()).build();
            }else{
                adRequest = new PublisherAdRequest.Builder().build();
            }

            //Si esta cargado se muestra, sino, se carga
            if (!interstitial.isLoaded()) {
                interstitial.loadAd(adRequest);
            } else {
                interstitial.show();
            }
        }
    });
}
```

En modo de pruebas se añade el id del dispositivo mediante un objeto tipo **TelephonyManager**.

Se comprueba que no esté cargado para lanzar una petición y cargarlo, y si no, se muestra.

Hay que tener en cuenta que el cargar el anuncio lleva un tiempo, así que no se mostrará nada más lanzar el método, habrá un delay de unos 2 o 3 segundos.

Un ejemplo de este tipo de anuncio sería:



## 7. TextureAtlas

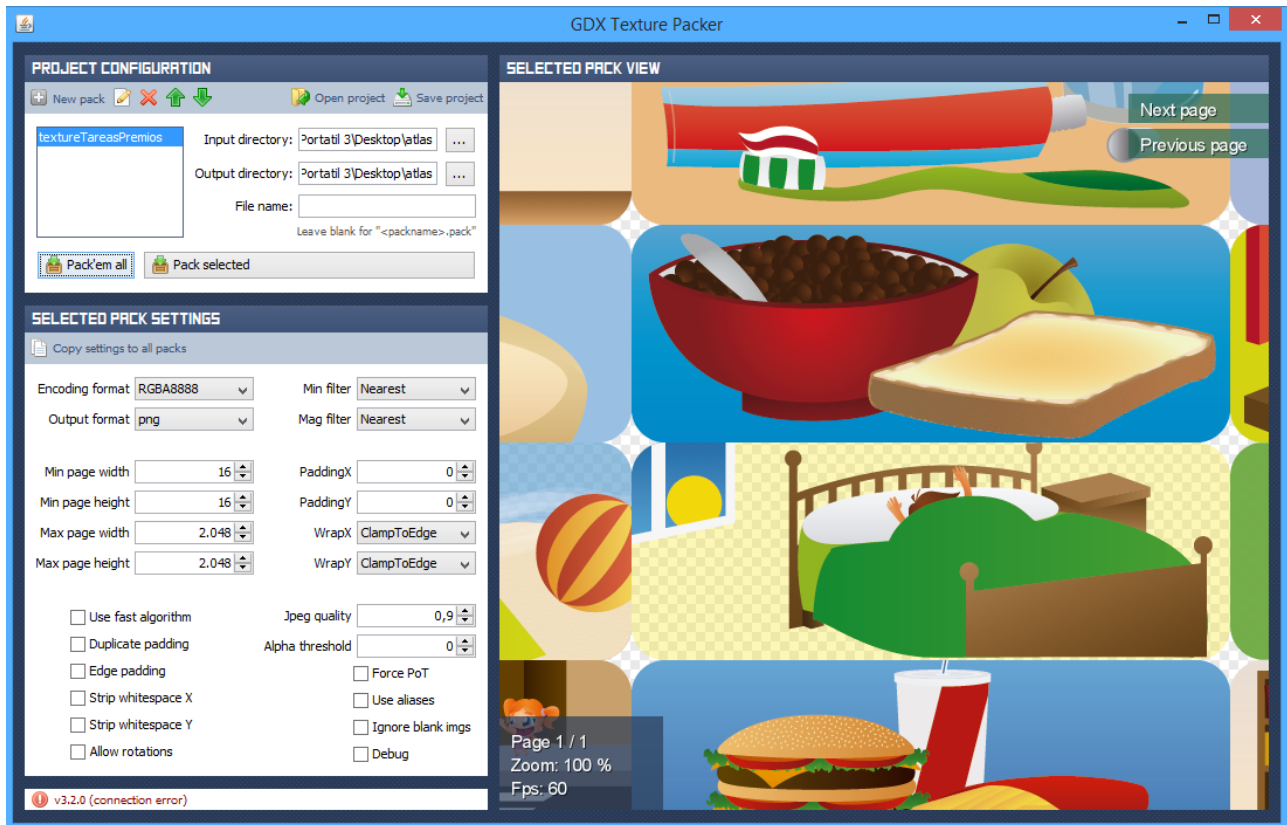
Inicialmente la interfaz de la aplicación fue generada con imágenes individuales (las cuales tienen que respetar que el alto y el ancho de la imagen en píxeles sean potencia de dos, requisito de libGDX), las cuales se cargan como Texture y de ahí extraíamos una región (TextureRegion). En el proceso final de depuración de la aplicación, nos dimos cuenta que tener tanta cantidad de imágenes sueltas requería de muchos recursos.

Se realizó una investigación y en estos casos los desarrolladores utilizan un TextureAtlas (mapa de imágenes, una imagen mayor que contiene todas las imágenes necesarias).



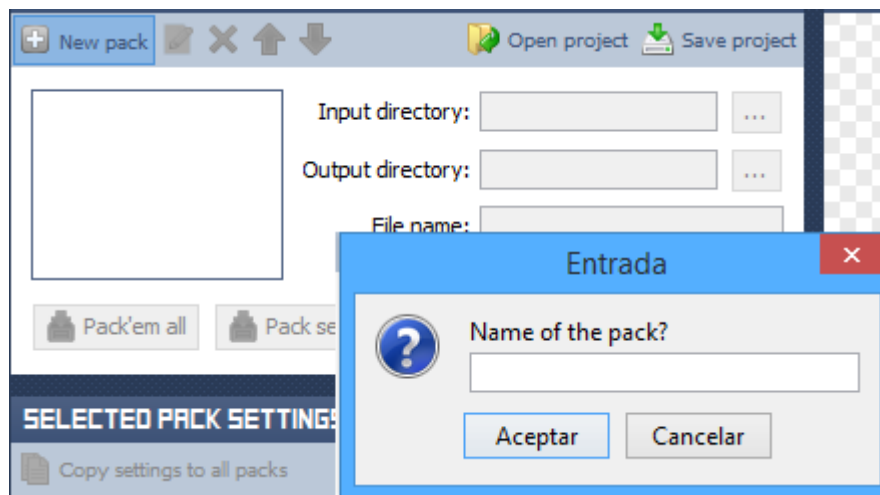
Para la realización del atlas existen distintos tipos de software, nosotros nos decantamos por uno exclusivo para libGDX (GUI Texture Packer libGDX).





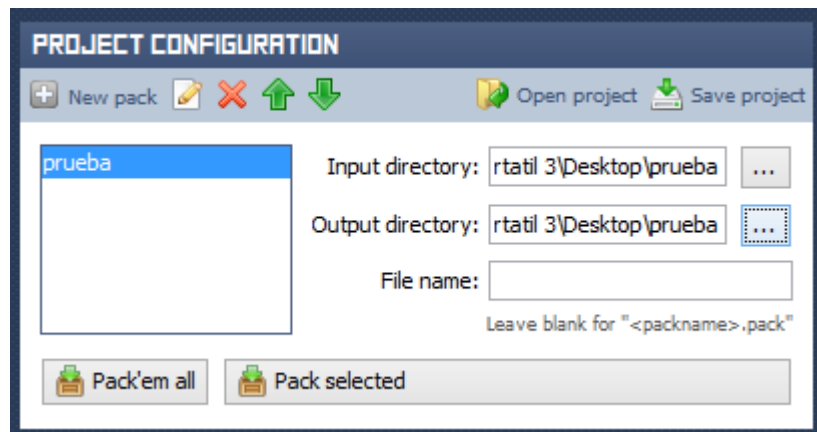
Ejemplo de uso de TexturePacker:

En primer lugar pulsamos New Pack e introducimos el nombre deseado para nuestro paquete:

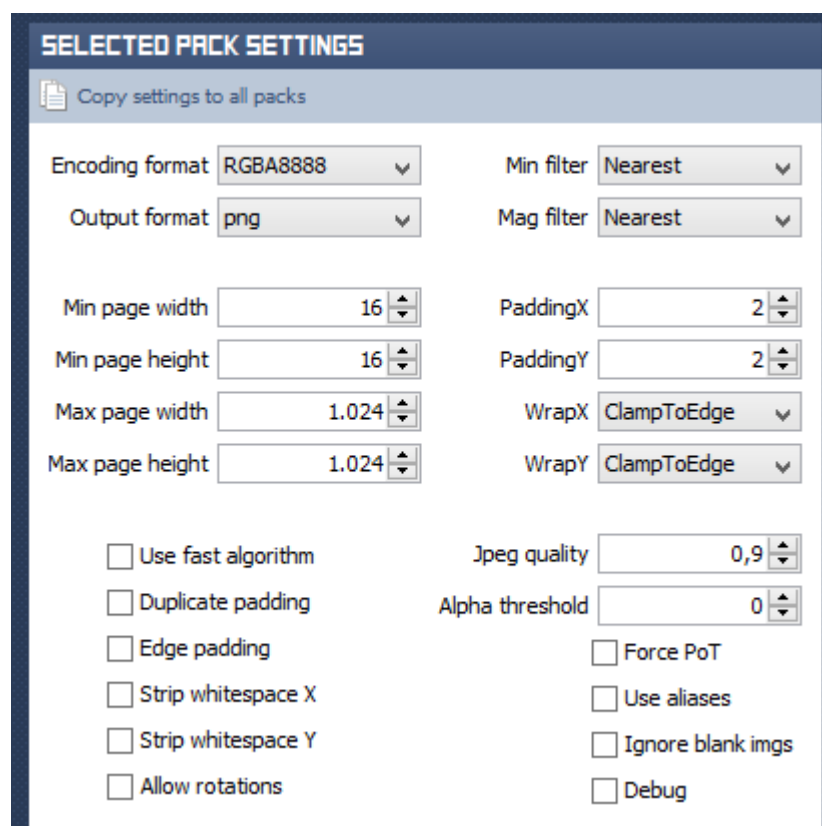


A continuación seleccionamos Input y Output directory desde donde TexturePacker coge automáticamente todas las imágenes que contenga, y donde se guardara nuestro paquete generado:





A continuación seleccionamos la configuración de creación de nuestro paquete:

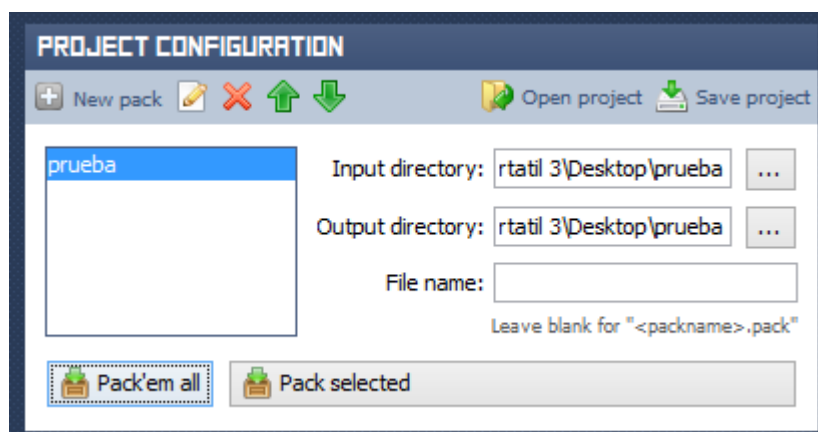


Explicación de cada opción de la configuración:

- **PaddingX:** El número de píxeles entre imágenes en el eje x.
- **PaddingY:** El número de píxeles entre imágenes en el eje y.
- **Edge Padding:** Si está marcado, la mitad del paddingX y paddingY se usa alrededor de los bordes de la textura de relleno.
- **Allow rotations:** Si está marcado, TexturePacker intentará empaquetar de la manera más eficiente, con la posibilidad de rotar las imágenes 90 grados. Se debe tener especial cuidado para dibujar estas regiones correctamente.
- **Min page Width:** Anchura mínima.
- **Min page Height:** Altura mínima.

- **Max page Width:** Anchura máxima.
- **Max page Height:** Altura máxima.
- **Strip Whitespace X:** Si está marcado, serán removidos píxeles en blanco en los bordes izquierdo y derecho de las imágenes de entrada. Se debe tener especial cuidado para dibujar estas regiones correctamente.
- **Strip Whitespace Y:** Si está marcado, serán removidos píxeles en blanco en los bordes superior e inferior de las imágenes de entrada. Se debe tener especial cuidado para dibujar estas regiones correctamente.
- **Alpha Threshold:** De 0 a 255. Valores alfa por debajo de este parámetro se tratan como cero cuando se elimina los espacios en blanco.
- **Min filter:** El filtro de reducción de la textura. Recomendado usar Nearest.
- **Mag filter:** El filtro de ampliación para la textura. Recomendado usar Nearest.
- **Wrap X:** Ajuste de envoltura en el eje x de la textura.
- **Wrap Y:** Ajuste de envoltura en el eje y de la textura.
- **Encoding format:** Formato con el que se guardara la textura, valores RGBA8888 (con canal alfa y mayor calidad), RGBA4444 (con canal alfa y menos calidad de color), etc.
- **Output Format:** Tipo de imagen con el que se guardara, jpg o png.
- **Jpeg Quality:** De 0 a 1. Ajuste de calidad si la salida es jpg.
- **Ignore Blank Images:** Si está marcado, TexturePacker no agregara regiones para las imágenes completamente en blanco.

Una vez configurado según preferencias, empaquetamos (Pack'em all):



Automáticamente se nos generaran varios archivos en el directorio de salida, un .pack que contiene todos los datos de nuestro Atlas (nombre, tamaño, posición de cada imagen) y uno o varios .png según sean necesarios.

Una vez generado nuestro Atlas, cargamos el .pack, el cual según su configuración accederá al .png asociado para extraer las imágenes.

```
/* Se carga el texture atlas */  
textureTareasPremios = new TextureAtlas(  
    (Gdx.files.internal(Assets.Texture_Atlas_Pack))) ;
```

Finalmente seleccionamos las imágenes incluidas en nuestro TextureAtlas mediante los distintos TextureRegion requeridos para su uso en la aplicación.

El método para extraer una región del Atlas es:

**nombreTextureAtlas.findRegion("imagen");**

Le indicaremos el nombre de la imagen que deseamos obtener en nuestro TextureRegion.

```
String imagePath = "tarea" + i + "";  
TextureRegion aux_region = new TextureRegion(babyeduca.textureTareasPremios.findRegion(imagePath));
```

## 8. Scroll Táctil

Durante varias ocasiones de la aplicación, requerimos del uso de un Scroll, al listar las tareas, premios, niños creados... Para ello libGDX nos facilita la clase de ScrollPane y Table. Con la que formamos una tabla de elementos para incluirla en nuestro Scroll.

Veamos un ejemplo:

En el caso de la actualización de tareas para un niño seleccionado, tenemos un método, el cual, le pasamos el niño en cuestión y en primer lugar comprueba que el Scroll no este vacío, para en este caso limpiarlo.

```
private void onUpdateTareas(int kid) {  
    // Si ya se creo el scroll del premio, se limpia para no sobreponer  
    // varios scrolls  
    if (scrollTasks != null)  
        scrollTasks.clear();
```

A continuación si el niño tiene tareas, las añadiremos a una tabla:

```
// Si el niño tiene tareas, se cargan
if (babyeduca.usuario.getKids().get(kid).getTasks().size() > 0) {

    // Añadiremos las tareas en una tabla, y la tabla en el scroll
    table_task_selected = new Table();
    table_task_selected.debug();
    table_task_selected.align(Align.top);
    for (int i = 0; i < babyeduca.usuario.getKids().get(kid).getTasks().size(); i++) {

        ImageLongTouch imageTarea = new ImageLongTouch(babyeduca.cache
            .getKids().get(kid).getTasks().get(i).getImage(), this,
            i, true);

        imageTarea.setWidth(babyeduca.anchura * 0.24f);
        imageTarea.setHeight((babyeduca.anchura * 0.24f) / 2.74f);
        /*
         * Se añade la tarea a la celda de la tabla
         */
        table_task_selected.add(imageTarea)
            .height(imageTarea.getHeight())
            .width(imageTarea.getWidth())
            .spaceBottom(babyeduca.altura * 0.04f);
        // Se añade una fila nueva
        table_task_selected.row();
    }
}
```

Finalmente añadiremos nuestra Tabla al Scroll. Le indicamos posición y tamaño dinámico que ocupara en pantalla, así como la velocidad y distancia de retorno.

```
scrollTasks = new ScrollPane(table_task_selected);
scrollTasks.setWidth(babyeduca.anchura * 0.24f);
scrollTasks.setHeight(babyeduca.altura * 0.60f);
scrollTasks.setPosition(babyeduca.anchura * 0.318f,
    babyeduca.altura * 0.13f);
//Se indica la altura del desplazamiento del Scroll, velocidad minima y maxima.
scrollTasks.setupOverscroll(babyeduca.altura * 0.23f, 200, 500);
stage.addActor(scrollTasks);
```

Como podéis observar en el código se creó una Clase ImageLongTouch. La función de esta Clase es otorgar a cada elemento de nuestros Scroll's (niños, tareas y/o premios) el pulsado largo para su posible eliminación.

```
ImageLongTouch imageTarea = new ImageLongTouch(babyeduca.cache
    .getKids().get(kid).getTasks().get(i).getImage(), this,
    i, true);
```

La clase ImageLongTouch extiende de Image y le pasamos una serie de parámetros para identificar la posición en concreto, si es una tarea o no, la imagen y el Listener necesario para el borrado de datos en el objeto.

```
/**
 * Imágenes con el evento de pulsado largo
 * @author Juan, Ivan
 *
 */
public class ImageLongTouch extends Image {

    /** Contador de segundos */
    SecondsCounter myCounter;

    /**
     * Image personalizada con listener de pulsado largo
     *
     * @param region
     *         TextureRegion de la imagen, será de premios y tareas
     * @param odk
     *         Listener para el borrado de objetos
     * @param position
     *         Posición del objeto en su array list
     * @param isTask
     *         Para saber si es una tarea o no
     */
    public ImageLongTouch(TextureRegion region, final OnDeleteObjects odk,
        int position, boolean isTask) {
        super(region);
    }
}
```

A continuación Construimos el contador de tiempo, y lo iniciamos o detenemos mediante las opciones del InputListener: touchDown y touchUp respectivamente.

```
//Se construye el contador
myCounter = new SecondsCounter(odk, position, false, isTask);

addListener(new InputListener() {

    public boolean touchDown(InputEvent event, float x, float y,
        int pointer, int button) {
        myCounter.startCount();

        return true;
    }

    public void touchUp(InputEvent event, float x, float y,
        int pointer, int button) {
        myCounter.stopCount();
    }
});
```

Dicho contador inicializa un Timer que controla internamente para cuando sea igual a 2 segundos nos lanza el dialogo de eliminación del elemento indicado, según sea niño, tarea o premio.

```
class Contador extends TimerTask {
    public void run() {
        seconds++;
        //Cuando llegue a 2 segundo, lanza el dialogo correspondiente
        if(seconds==2){
            if(isKid){
                okd.showDeleteKidDialog(position);
            }else{
                okd.showDeleteTaskRewardDialog(position, isTask);
            }
            //Se para el contador
            this.cancel();
        }
        System.out.println("segundo: " + seconds);
    }
}
```

Este dialogo se lanza mediante la llamada a los métodos de la interfaz OnDeleteObjects:

```
package com.neurodigital.babyeduca.interfaces;
/**
 * Interface relacionada con el borrado de niños, tareas y premios.
 * Mostrar sus respectivos dialogos de confirmacion
 * @author Juan, Ivan
 */
public interface OnDeleteObjects {
    /**
     * Muestrar el dialogo de borrado del niño seleccionado
     * @param kid_chosen Niño seleccionado
     */
    void showDeleteKidDialog(int kid_chosen);
    /**
     * Muestra el dialogo de la tarea o premio seleccionado
     * @param objectSelected Tarea o premio seleccionado
     * @param isTask Para saber si es tarea o premio
     */
    void showDeleteTaskRewardDialog(int objectSelected, boolean isTask);
}
```

En nuestra clase principal sobrescribimos dichos métodos para lanzar en cada caso el dialogo en cuestión, que simplemente nos pedirá una confirmación de la eliminación.



```
@Override
public void showDeleteKidDialog(int kid_chosen) {
    babyeduca.state = State.DIALOG;
    onDrawWindowDeleteKid(kid_chosen);
    // Se quitan los listeners de los botones de la ventana anterior
    disableListeners();
    stage.addActor(windowDeleteKid);
    isDeleteKidDialogShown = true;
}

@Override
public void showDeleteTaskRewardDialog(int objectSelected, boolean isTask) {
    babyeduca.state = State.DIALOG;
    onDrawWindowDeleteTaskReward(objectSelected, isTask);
    // Se quitan los listeners de los botones de la ventana anterior
    disableListeners();
    stage.addActor(windowDeleteTaskReward);
}
}
```

En algunos dispositivos móviles nos surgió un problema, algunas pantallas táctiles son más sensibles que otras a la hora del pulsado-deslizamiento, vimos que hay una constante fija para cuando se está realizando scrolling (Eje y de coordenadas), con lo que en el evento touchUp de cada imagen comprobamos que nuestra posición en la coordenada y sea diferente a dicha constante, en ese caso, realizamos las opciones requeridas.

```
public class DefaultReward extends Image {
    /**Constante que devuelve cuando hace scrolling */
    int SCROLLING_CONSTANT = -2147483648;
```

```
public void touchUp(InputEvent event, float x, float y,
    int pointer, int button) {
    System.out.println("Segunda Y "+(int)y);
    //Cuando no haga scrolling, añadimos el objeto
    if ((int)y!=SCROLLING_CONSTANT) {
        game.usuario.getKids().get(game.kidChosen)
            .addReward(new Reward(imagePath, 1));
    }
}
```

## 9. MANUAL DE CONFIGURACIÓN Y FUNCIONAMIENTO DE LA APLICACION

La aplicación no requiere de ninguna configuración previa, solo bastara con la descarga e instalación de la aplicación en nuestro equipo/dispositivo desde las diferentes tiendas Online para nuestro sistema. Tiendas para cada sistema:

- Android: Google Play Store.
- ios: APP Store.

## 10. MANUAL DE USUARIO

[Enlace al manual de usuario de BabyEduca](#)

## 11. PLAN DE FORMACION A LOS USUARIOS DE LA APLICACIÓN

### 11.1. *Contenidos a tratar*

No hará falta un plan de formación propiamente dicho para el uso de BabyEduca. Simplemente bastará con ver el manual de usuario y/o el video explicativo de la aplicación. En ellos están destacados los aspectos más relevantes que la aplicación en sí lleva a cabo.

Es una aplicación sencilla de usar e intuitiva, por lo que no habrá problemas en su uso.

### 11.2. *Tiempo estimado de impartición de cada contenido*

El tiempo estimado será de entre 15 y 30 minutos, lo suficiente para que el usuario se familiarice con la aplicación y coja soltura en su uso.

Hemos hecho pruebas con usuarios que no conocían la aplicación y no disponían de conocimiento alguno y el resultado fue el esperado.



## 12. BIBLIOGRAFÍA Y FUENTES DE INFORMACION

- **Libro conceptos básicos sobre LibGdx:**
  - PFM\_AprendeProgramarVideojuegosAndroid.pdf
  - <https://github.com/libgdx/libgdx/wiki>
- **Primeros pasos con LibGdx (Tutorial de ejemplo):**
  - <http://libgdxspain.blogspot.com.es/>
- **Creación de fuentes .FNT .PNG:**
  - <http://www.angelcode.com/products/bmfont/>
- **Ejemplo de uso, clase Calendar:**
  - <http://www.lawebdelprogramador.com/codigo/Java/2299-Ejemplo-de-la-utilizacion-de-la-clase-Calendar.html>
- **Elaboración de Nine Patch Images:**
  - <http://libgdx tutorials.blogspot.com.es/2013/09/libgdx-tutorial-9-nine-patch-images.html>
- **Uso de BitmapFont:**
  - <http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/g2d/BitmapFont.html>
- **Vídeo Tutoriales LibGdx:**
  - <https://www.youtube.com/playlist?list=PLXY8okVWvwZ0JOwHiH1TntAdq-UDPnC2L>
- **Javadoc de LibGdx:**
  - <http://libgdx.badlogicgames.com/nightlies/docs/api/>
- **Controlar fecha:**
  - <http://www.chilecomparte.cl/topic/2086561-validar-fechas-orientado-a-objetos-java/>
  - <http://www.javerosanonimos.com/2012/04/calcular-edad-con-fecha-de-nacimiento.html>
- **Creación de Scroll:**
  - <http://stackoverflow.com/questions/15484077/libgdx-and-scrollpane-with-multiple-widgets>
- **Uso de la WebCam (PC y Android):**
  - <http://webcam-capture.saxos.pl/>
  - <http://developer.android.com/training/camera/photobasics.html>
- **Galería (Android):**
  - <http://viralpatel.net/blogs/pick-image-from-galary-android-app/>

- **FileChooser (Escritorio):**
  - <http://docs.oracle.com/javase/7/docs/api/javax/swing/JFileChooser.html>
- **Admob (Publicidad):**
  - <https://developers.google.com/mobile-ads-sdk/docs/?hl=es>
- **Creación de frame para primer plano de jfilechooser:**
  - <http://www.java-gaming.org/index.php?;topic=28923.0>
- **Dudas:**
  - <http://stackoverflow.com/>
- **Antialiasing**
  - [http://en.wikipedia.org/wiki/Multisample\\_anti-aliasing#Sampling\\_methods](http://en.wikipedia.org/wiki/Multisample_anti-aliasing#Sampling_methods)
- **Generacion y uso TextureAtlas:**
  - <http://bitiotic.com/blog/2012/05/10/tutorial-for-texturepacker-and-libgdx/>
  - <https://github.com/libgdx/libgdx/wiki/Texture-packer>
  - <http://libdgxtutorials.blogspot.com.es/2013/09/libgdx-tutorial-8-using-texture-packer.html>