Interpool

Modelo de Diseño

Versión 10.1

Historia de revisiones

|  |  |  |  |
| --- | --- | --- | --- |
| Fecha | Versión | Descripción | Autor |
| 03/09/10 | 1.0 | Creación del documento | Marcos Sander |
| 19/09/10 | 6.0 | Cambio del diseño de iniciar iteración | Marcos Sander |
| 19/09/10 | 6.1 | Revisión de SQA | Javier Madeiro |
| 05/10/10 | 9.0 | Se agregan diseños de varios casos de uso y sesiones faltantes. | Marcos Sander |
| 09/10/10 | 9.1 | Revisión de SQA | Javier Madeiro |
| 16/10/10 | 10.0 | Se agregó el diagrama de clases y las interfaces de los subsistemas. | Marcos Sander |
| 17/10/10 | 10.1 | Revisión de SQA | Javier Madeiro |

Contenido

[Contenido 2](#_Toc275047815)

[1. Introducción 4](#_Toc275047816)

[1.1. Propósito 4](#_Toc275047817)

[1.2. Alcance 4](#_Toc275047818)

[1.3. Referencias 4](#_Toc275047819)

[1.4. Visión general 4](#_Toc275047820)

[2. Diseño de Casos de Uso 5](#_Toc275047821)

[2.1. Iniciar Iteración 5](#_Toc275047822)

[2.1.1. Diagrama de paquetes 5](#_Toc275047823)

[2.1.2. Diagrama de Interacción 6](#_Toc275047824)

[2.1.3. Diseño de Flujo de eventos 6](#_Toc275047825)

[2.1.4. Requerimientos especiales o de implementación 6](#_Toc275047826)

[2.2. Interrogar personaje 7](#_Toc275047827)

[2.2.1. Diagrama de paquetes 7](#_Toc275047828)

[2.2.2. Diagrama de Interacción 7](#_Toc275047829)

[2.2.3. Diseño de Flujo de eventos 8](#_Toc275047830)

[2.2.4. Requerimientos especiales o de implementación 8](#_Toc275047831)

[2.3. Obtener ciudades 9](#_Toc275047832)

[2.3.1. Diagrama de paquetes 9](#_Toc275047833)

[2.3.2. Diagrama de Interacción 10](#_Toc275047834)

[2.3.3. Diseño de Flujo de eventos 10](#_Toc275047835)

[2.3.4. Requerimientos especiales o de implementación 10](#_Toc275047836)

[2.4. Viajar 11](#_Toc275047837)

[2.4.1. Diagrama de paquetes 11](#_Toc275047838)

[2.4.2. Diagrama de Interacción 12](#_Toc275047839)

[2.4.3. Diseño de Flujo de eventos 12](#_Toc275047840)

[2.4.4. Requerimientos especiales o de implementación 12](#_Toc275047841)

[2.5. Filtrar sospechosos 13](#_Toc275047842)

[2.5.1. Diagrama de paquetes 13](#_Toc275047843)

[2.5.2. Diagrama de Interacción 14](#_Toc275047844)

[2.5.3. Diseño de Flujo de eventos 14](#_Toc275047845)

[2.5.4. Requerimientos especiales o de implementación 14](#_Toc275047846)

[2.6. Emitir orden de arresto 15](#_Toc275047847)

[2.6.1. Diagrama de paquetes 15](#_Toc275047848)

[2.6.2. Diagrama de Interacción 15](#_Toc275047849)

[2.6.3. Diseño de Flujo de eventos 16](#_Toc275047850)

[2.6.4. Requerimientos especiales o de implementación 16](#_Toc275047851)

[2.7. Actualizar datos ciudades 17](#_Toc275047852)

[2.7.1. Diagrama de paquetes 17](#_Toc275047853)

[2.7.2. Diagrama de Interacción 18](#_Toc275047854)

[2.7.3. Diseño de Flujo de eventos 18](#_Toc275047855)

[2.7.4. Requerimientos especiales o de implementación 18](#_Toc275047856)

[2.8. Actualizar datos famosos 19](#_Toc275047857)

[2.8.1. Diagrama de paquetes 19](#_Toc275047858)

[2.8.2. Diagrama de Interacción 19](#_Toc275047859)

[2.8.3. Diseño de Flujo de eventos 20](#_Toc275047860)

[2.8.4. Requerimientos especiales o de implementación 20](#_Toc275047861)

[3. Diseño de Objetos 21](#_Toc275047862)

[4. Diseño de Subsistemas 22](#_Toc275047863)

[4.1. Subsistemas Específicos 22](#_Toc275047864)

[4.1.1. Service Subsystem 22](#_Toc275047865)

[4.1.2. Bussines Subsystem 24](#_Toc275047866)

[4.1.3. ExternServices Subsystem 26](#_Toc275047867)

[4.1.4. Data Subsystem 28](#_Toc275047868)

[5. Revisión de la Interfaz de Usuario 29](#_Toc275047869)

[5.1. Formatos individuales de interfaz de pantalla 29](#_Toc275047870)

[5.2. Controles y elementos de diseño de interfaz de pantalla 29](#_Toc275047871)

[5.3. Forma de navegación de interfaz de pantalla 29](#_Toc275047872)

[5.4. Formatos de impresión 30](#_Toc275047873)

[6. Diseño de Datos 31](#_Toc275047874)

[6.1. Especificación de la Distribución de Datos 31](#_Toc275047875)

1. Introducción

En este documento se presenta el diseño del sistema a construir.

* 1. Propósito

Este documento tiene como fin ayudar a los implementadores del sistema a construirlo. La audiencia esperada del documento es del equipo de implementación del sistema.

* 1. Alcance

Es influenciado fuertemente por la descripción de requerimientos del sistema y por el modelo de casos de uso del mismo, obtenidos al hacer el relevamiento de requerimientos. Impacta en la implementación del sistema.

* 1. Referencias

Modelo de casos de uso.

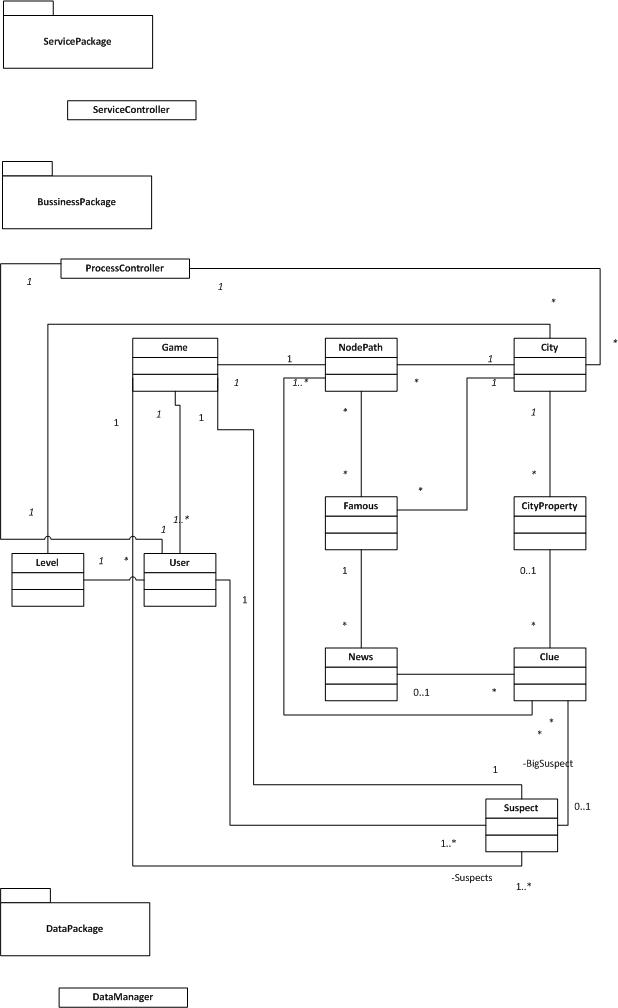
Descripción de Requerimientos del Sistema.

Descripción de la Arquitectura del Sistema.

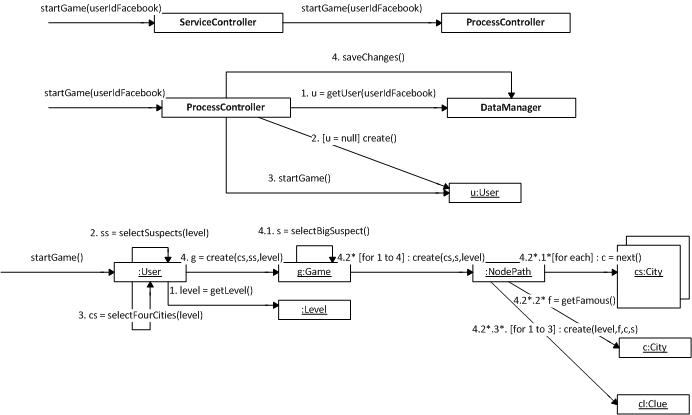
* 1. Visión general

El documento está organizado de la siguiente manera. En la sección 2 del mismo, se describe el diseño de los casos de uso del sistema. En la sección 3, el diseño de objetos del sistema. En la sección 4, el diseño de subsistemas. En la sección 5 se describe desde un punto de vista de diseño la interfaz que tendrá el sistema y en la sección 6 se describe el modelo de datos del mismo.

1. Diseño de Casos de Uso
   1. Iniciar Iteración
      1. Diagrama de paquetes



* + 1. Diagrama de Interacción



* + 1. Diseño de Flujo de eventos

Para la realización de este caso de uso, entran en juego una instancia de la clase ServiceController, que brinda los servicios al Windows Phone y la clase ProcessController, que emula la capa de procesamientos de los servicios del servidor que está en Azure. Con el servicio se le pasa un identificador del jugador (que es el mismo identificador utilizado en Facebook).

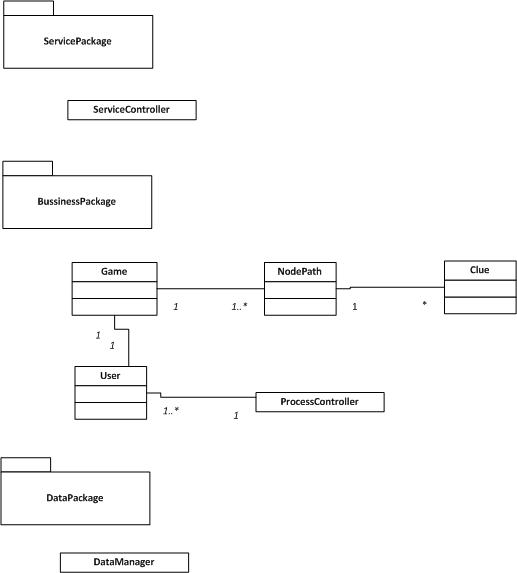
El ProcessController crea un nuevo juego. La instancia de juego, también es la responsable de generar los sospechosos, de los datos de los amigos del usuario que son traídos desde Facebook. Luego también la instancia de clase juego, genera el camino por el cual irá el gran sospechoso. Esta clase se comunicará con otras clases para obtener el resultado esperado (clase ciudad, clase famoso, clase noticia, …).

* + 1. Requerimientos especiales o de implementación

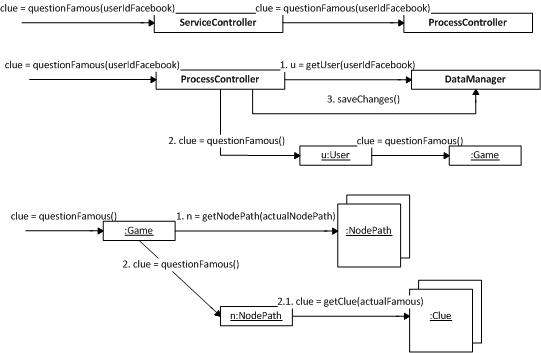
El usuario debe tener una cuenta Facebook para que pueda realizarse esta operación. También debe tener la cantidad de amigos necesaria, según el nivel de juego en que está el usuario, la lista de sospechosos será de un largo determinado.

También se tiene cómo restricción, que la ejecución de esta operación no deberá tener una duración mayor a 15 segundos.

* 1. Interrogar personaje
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



* + 1. Diseño de Flujo de eventos

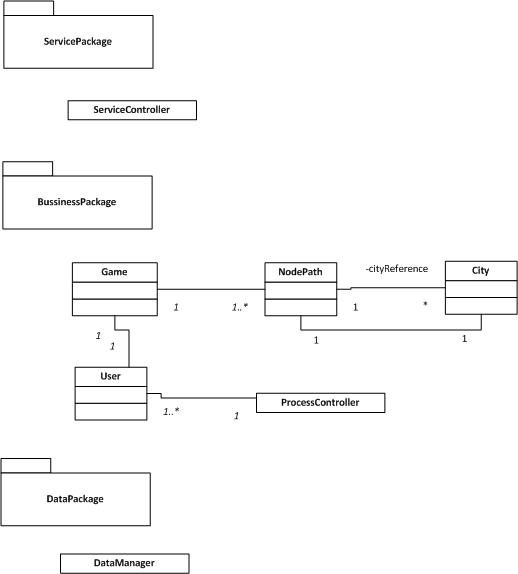
Al igual que para el caso de uso iniciar iteración y para todos los casos de uso del juego, la solicitud de interrogar al personaje le llega a la clase ServiceController, esta se comunica con ProcessController quién es la encargada de llevar a cabo la operación.

El ProcessController se comunica con una instancia de DataManager, para obtener desde la base de datos al usuario con el identificador de Facebook dado. Esta instancia de usuario se comunica con la instancia de juego asociada, que es la encargada de ver cuál es el nodo actual, y a ese nodo solicitarle la pista del famoso.

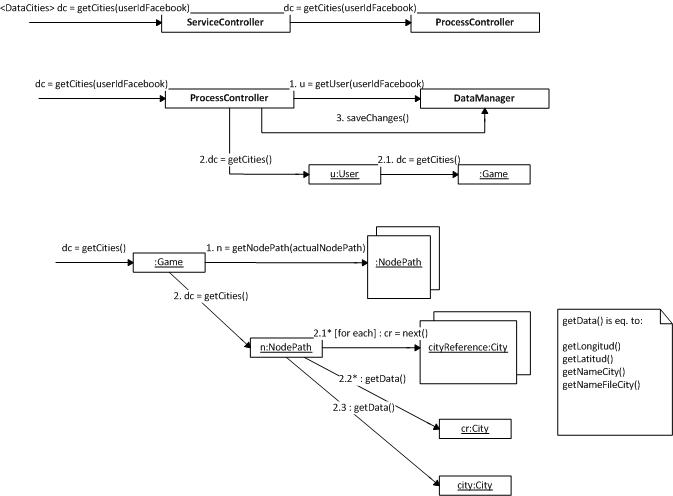
* + 1. Requerimientos especiales o de implementación

La pista está formada por una o ninguna pista de ciudad dinámica o estática, también por una o ninguna pista de sospechoso y por una o ninguna noticia del famoso que da la pista. La pista debe tener por lo menos uno de los elementos mencionados anteriormente y para un nodo dado estos elementos deben aparecer alguna vez, entre las pistas que se den en ese nodo.

* 1. Obtener ciudades
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



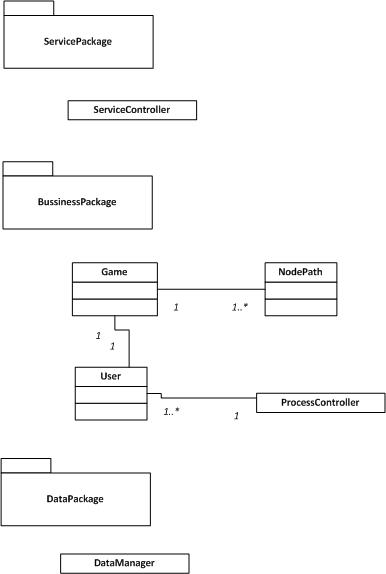
* + 1. Diseño de Flujo de eventos

El flujo de operaciones y de acciones es similar a las operaciones anteriores. Para destacar es que el NodePath actual de un juego dado, tiene una referencia a una ciudad, que es la ciudad actual en la que está el juego y 2 ciudades más, nombradas cómo cityReference, que son las 2 ciudades posibles donde podría haber viajado el jugador.

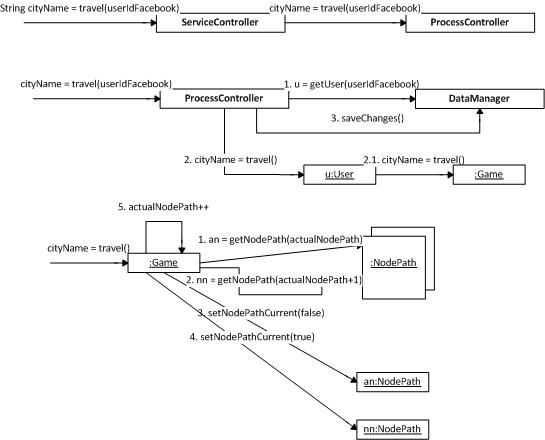
* + 1. Requerimientos especiales o de implementación

Las ciudades posibles de viaje y la ciudad actual deben ser distintas entre sí, y también todas ellas deben ser distintas a la ciudad actual del NodePath anterior.

* 1. Viajar
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



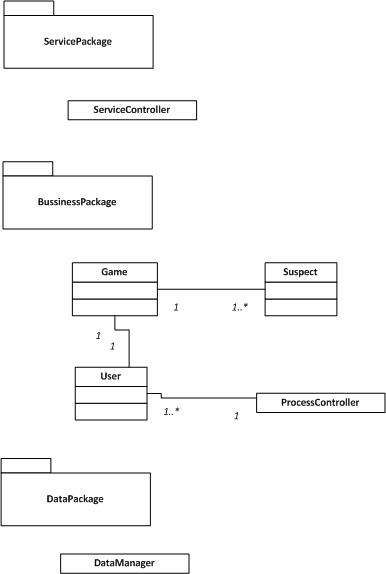
* + 1. Diseño de Flujo de eventos

La clase Game, es la responsable de ejecutar la lógica más importante para esta operación. Setea al NodePath actual que corresponde y borra esa bandera para el NodePath actual anterior.

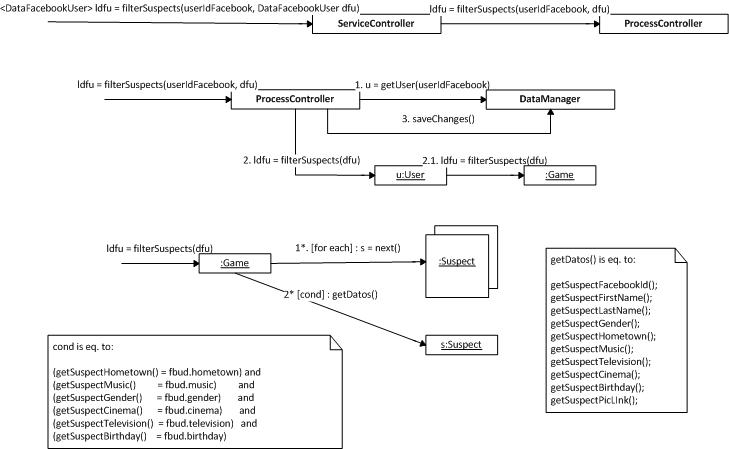
* + 1. Requerimientos especiales o de implementación

El usuario no debe estar en el último NodePath o última ciudad para poder viajar.

* 1. Filtrar sospechosos
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



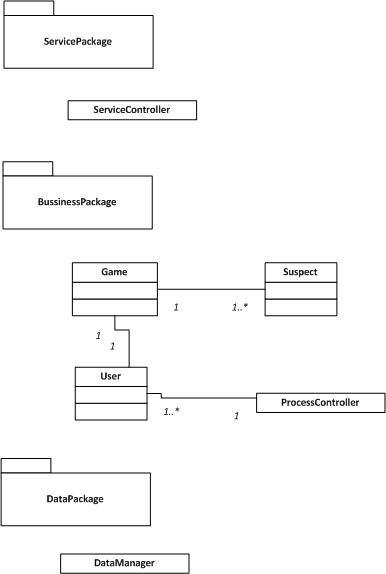
* + 1. Diseño de Flujo de eventos

La operación de filtrar sospechosos recibe como entrada el userIdFacebook para identificar al usuario que está jugando y también recibe un sdt DataFacebookUser que contiene los datos para filtrar. Se devuelve una lista de DataFacebookUser con los datos de cada sospechoso que pasó el filtro. Las clases participantes en la solución, son similares a las anteriores. La clase Game es la encargada de llevar a cabo la lógica para filtrar a los sospechosos de ese juego.

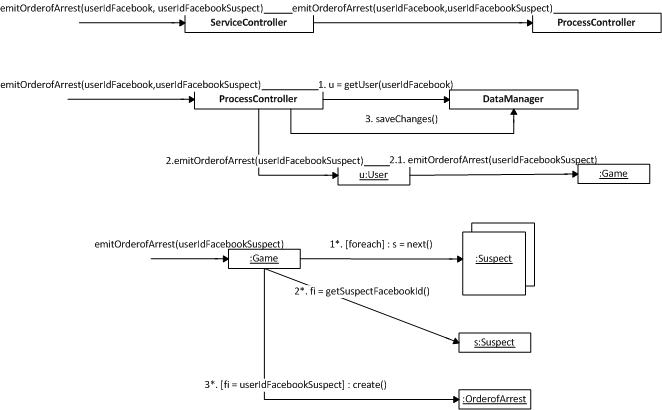
* + 1. Requerimientos especiales o de implementación

No tiene.

* 1. Emitir orden de arresto
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



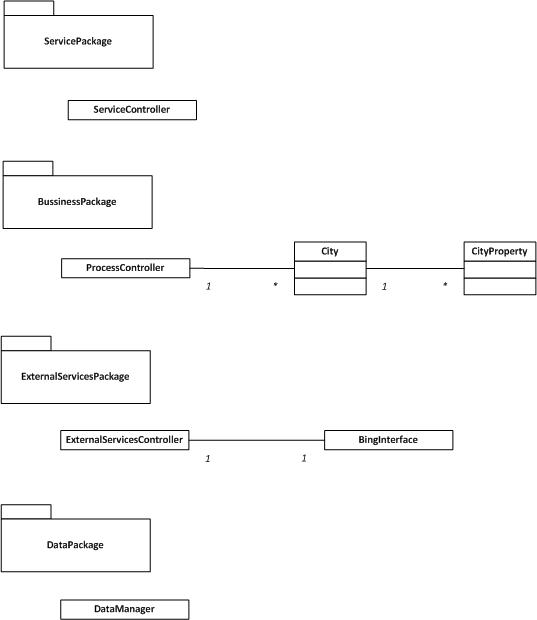
* + 1. Diseño de Flujo de eventos

La clase Game es la que tiene la responsabilidad de crear la orden de arresto. Se crea una instancia de la clase OrderofArrest, que indica que se realizó la orden de arresto correspondiente a este juego.

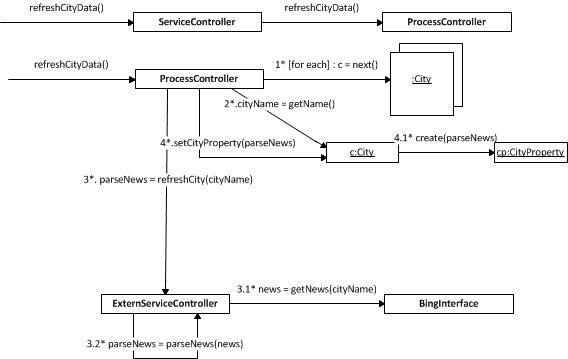
* + 1. Requerimientos especiales o de implementación

No tiene.

* 1. Actualizar datos ciudades
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



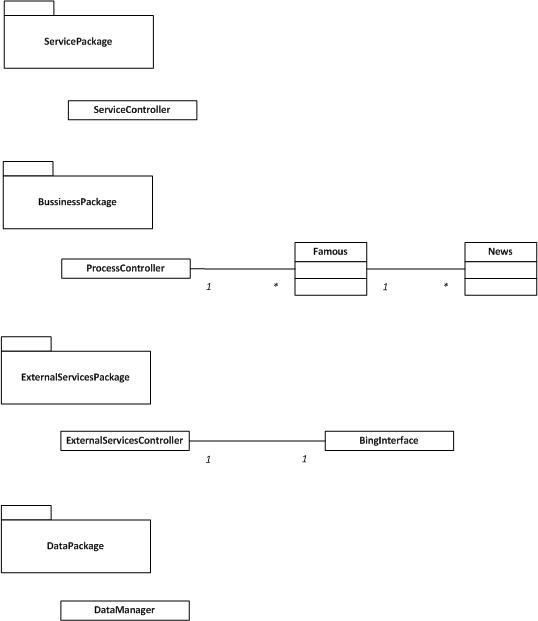
* + 1. Diseño de Flujo de eventos

Esta operación varía un poco la lógica y las responsabilidades respecto a las operaciones anteriores. La clase ProcessController es la encargada de ejecutar la lógica principal de la operación y una instancia de ExternServiceController es la encargada de actualizar los datos y parsearlos para que sean utilizables en el juego. Luego de la operación cada ciudad quedará con sus propiedades dinámicas actualizadas en la base de datos.

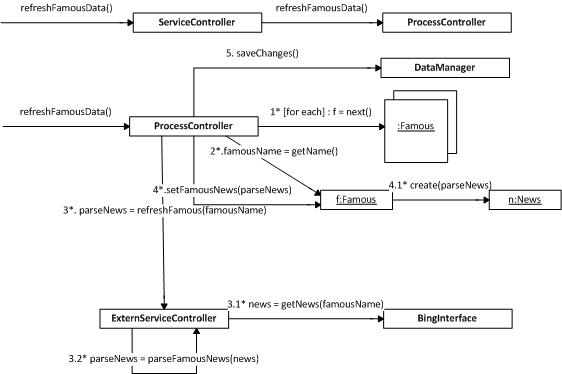
* + 1. Requerimientos especiales o de implementación

No tiene.

* 1. Actualizar datos famosos
     1. Diagrama de paquetes



* + 1. Diagrama de Interacción



* + 1. Diseño de Flujo de eventos

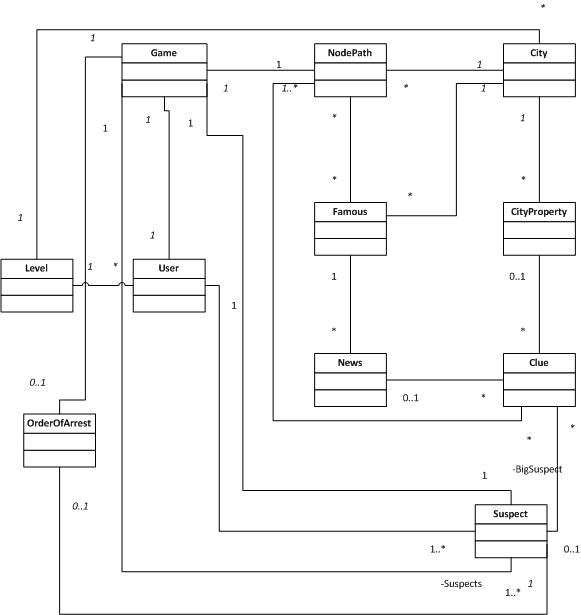
La lógica de la operación es similar a la operación ActualizarDatosCiudades. Se actualizan los datos para cada famoso. Se crean nuevas instancias de News, que contienen la noticia parseada, o sea lo que va a “decir” el famoso cuando dé pistas.

* + 1. Requerimientos especiales o de implementación

No tiene.

1. Diseño de Objetos

Se presenta a continuación el diagrama de clases con las clases correspondientes al Bussiness Package:



1. Diseño de Subsistemas
   1. Subsistemas Específicos
      1. Service Subsystem

**Propósito**

El subsistema de servicios es el encargado **de la comunicación del Windows Phone con Azure**. Entre otras cosas provee la interfaz por la cual se comunicará el Phone con Azure, y también en este subsistema se implementará la lógica para que el juego pueda ser jugado por muchos usuarios a la vez.

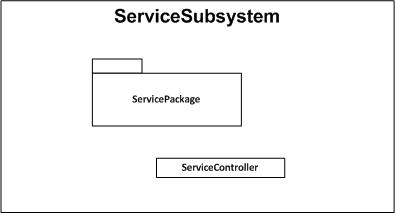
La comunicación Phone-Azure, se realiza mediante Windows Comunication Services.

**Función**

Este subsistema básicamente **recibe** y **ordena** los pedidos de parte del Phone, para que sean procesados luego por el subsistema de negocios que está en el servidor.

**Subordinados**

El siguiente diagrama muestra los objetos de diseño y subsistemas de diseño que componen a este subsistema:



**Dependencias**

|  |  |  |
| --- | --- | --- |
| Subsistema del que depende | Naturaleza de interacción | Características |
| Bussines Subsystem | Por cada servicio que se ejecuta en este subsistema, se hace una llamada al subsistema de negocios. | Para cada servicio que provee el servidor al Phone, el subsistema de negocios ejecuta la lógica necesaria para poder llevar a cabo la tarea. |

**Recursos**

No tiene dependencias con recursos externos al sistema de diseño.

**Interfaces**

|  |  |  |
| --- | --- | --- |
| Interacción | Método de interacción | Reglas de la interacción |
| IServices | void StartGame(string userIdFacebook); | El método empieza un nuevo juego.  userIdFacebook, es el id del usuario en Facebook. |
|  | DataCity GetCurrentCity(string userIdFacebook); | El método devuelve la ciudad actual del juego, en que está parado el usuario.  DataCity, es un tipo de datos que contiene los datos de la ciudad  userIdFacebook, es el id del usuario en Facebook. |
|  | List<DataCity> GetPossibleCities(string userIdFacebook); | Devuelve las ciudades posibles a viajar.  List<DataCity>, es una lista de las ciudades.  userIdFacebook, es el id del usuario de Facebook. |
|  | DataFamous GetCurrentFamous(string userIdFacebook, int numClue); | Devuelve al famoso actual, al que se va a interrogar.  DataFamous, contiene los datos del famoso  userIdFacebook, id del usuario de Facebook.  numClue, es el orden del famoso al que se va a interrogar. |
|  | List<DataFacebookUser> FilterSuspects(string userIdFacebook, DataFacebookUser fbud); | Devuelve la lista de usuarios filtrados según el filtro pasado por parámetro.  List<DataFacebookUser>, es la lista de usuarios filtrados  userIdFacebook, id del usuario en Facebook.  Fbud, contiene los valores del filtro. |
|  | void EmitOrderOfArrest(string userIdFacebook, string userIdFacebookSuspect); | Este método emite la orden de arresto para un sospechoso dado pasado por parámetro.  userIdFacebook, id del usuario en Facebook  userIdFacebookSuspect, es el id del sospechoso a emitir la orden de arresto. |
|  | DataCity Travel(string userIdFacebook, string nameNextCity); | Realiza el viaje desde la ciudad actual hasta la nueva ciudad, devuelve los datos de la ciudad a viajar en caso que sea correcto el viaje, sino devuelve los datos de la ciudad actual.  userIdFacebook, es el id del usuario en Facebook.  nameNextCity, es el nombre de la ciudad a la que se quiere viajar. |
|  | DataClue GetClueByFamous(string userIdFacebook, int numFamous); | Se obtiene la pista que la da el famoso al que se va a interrogar.  userIdFacebook, es el id del usuario en Facebook.  numFamous, es el orden del famoso a interrogar. |

* + 1. Bussines Subsystem

**Propósito**

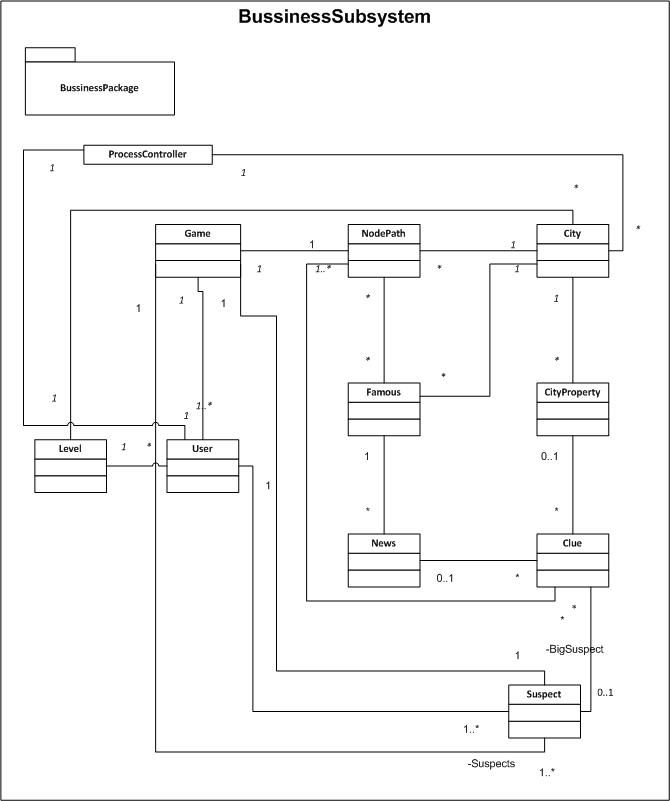
El subsistema de negocios es el encargado **de la realización de los servicios brindados por Azure**. Este subsistema recibe pedidos de ejecución de servicios del subsistema de Servicios y los ejecuta, obteniéndose un resultado dado en el sistema.

**Función**

Este subsistema **ejecuta la lógica** para poder **realizar los servicios** que son solicitados por el subsistema de servicios.

**Subordinados**

El siguiente diagrama muestra los objetos de diseño y subsistemas de diseño que componen a este subsistema:



**Dependencias**

|  |  |  |
| --- | --- | --- |
| Subsistema del que depende | Naturaleza de interacción | Características |
| Service Subsystem | Cada servicio que se ejecuta en este subsistema, es en respuesta a un servicio solicitado por el subsistema de servicios. | Para cada servicio que provee el servidor al Phone, el subsistema de negocios ejecuta la lógica necesaria para poder llevar a cabo la tarea. |
| ExternalServices Subsystem | Cuando algunos servicios de este subsistema se necesitan comunicar con Facebook o Bing, lo hacen a través del subsistema de Servicios externos | Para cada servicio que realiza este subsistema, que tiene la necesidad de comunicarse con Facebook o Bing, el mismo ejecuta funciones del ExternalServices Subsystem para poder recibir la información necesaria para llevar a cabo el servicio dado. |
| Data Subsystem | Necesidad de consultar o guardar datos en la base de datos. | Cada vez que se ejecutan operaciones, se obtienen los datos necesarios desde la base y se guardan los cambios en la base de datos también. |

**Recursos**

No tiene dependencias con recursos externos al sistema de diseño.

**Interfaces**

|  |  |  |
| --- | --- | --- |
| Interacción | Método de interacción | Reglas de la interacción |
| [Objeto que implementa la interacción] |  |  |

* + 1. ExternServices Subsystem

**Propósito**

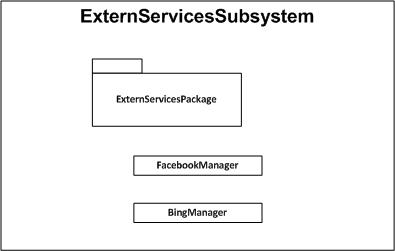
El subsistema de servicios externos es el encargado **de la comunicación de Azure con Facebook y Bing**. Provee una interfaz para que pueda comunicarse con el subsistema de negocios. Se comunica con Facebook y Bing a través de interfaces adecuadas en cada caso, y también posee la lógica para transformar los datos obtenidos de esos servicios externos en datos que puedan ser utilizados en el juego (por ejemplo se hace un parseo de las noticias acerca de las ciudades del juego, para obtener características dinámicas de cada ciudad).

**Función**

Sirve como medio de comunicación entre el subsistema de negocios y los subsistemas externos de Facebook y Bing.

**Subordinados**

El siguiente diagrama muestra los objetos de diseño y subsistemas de diseño que componen a este subsistema:



**Dependencias**

|  |  |  |
| --- | --- | --- |
| Subsistema del que depende | Naturaleza de interacción | Características |
| Bussines Subsystem | Cuando se ejecuta un servicio en el subsistema de negocios que tiene la necesidad de consultar datos desde Facebook o Bing. | Provee una interfaz al subsistema de negocios para que este pueda obtener los datos que necesita, al fin de poder llevar a cabo los servicios que implementa. |
| Data Subsystem | Necesidad de obtener o persistir datos en la base de datos | El subsistema de datos, persiste o recupera datos que han sido previamente guardados en la base de datos y que son requeridos por el subsistema de servicios externos. |

**Recursos**

Depende de los recursos externos Facebook y Bing.

**Interfaces**

|  |  |  |
| --- | --- | --- |
| Interacción | Método de interacción | Reglas de la interacción |
| [Objeto que implementa la interacción] |  |  |

* + 1. Data Subsystem

**Propósito**

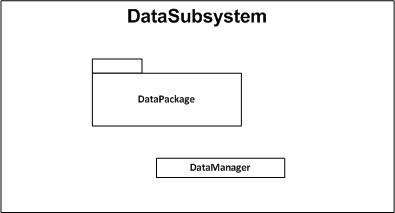
El subsistema de datos es el encargado **de la comunicación de Azure con la base de datos de Azure**. Provee los servicios y consultas que se pueden hacer a la base de datos, a fin de poder guardar, eliminar, consultar o modificar los datos del juego que están en la base de datos.

**Función**

Recibe desde los subsistemas de negocios y servicios externos los pedidos de guardar, eliminación, consulta y modificación de datos que están en la base de datos y realiza eso a través de consultas a la base de datos.

**Subordinados**

El siguiente diagrama muestra los objetos de diseño y subsistemas de diseño que componen a este subsistema:



**Dependencias**

|  |  |  |
| --- | --- | --- |
| Subsistema del que depende | Naturaleza de interacción | Características |
| Bussines Subsystem | Necesidad de datos que están en la base de datos, o de guardar, modificar o eliminar los mismos. | Ejecuta las consultas solicitadas por el subsistema, en la base de datos. |
| ExternServices Subsystem | Necesidad de datos que están en la base de datos o de guardar, modificar o eliminar los mismos. | Ejecuta las consultas solicitadas por el subsistema, en la base de datos. |

**Recursos**

Tiene dependencia con la base de datos de Azure (SQL Azure Database).

**Interfaces**

|  |  |  |
| --- | --- | --- |
| Interacción | Método de interacción | Reglas de la interacción |
| [Objeto que implementa la interacción] |  |  |

1. Revisión de la Interfaz de Usuario
   1. Formatos individuales de interfaz de pantalla

Cada pantalla será representada como un objeto del tipo PhoneApplicationPage.

* 1. Controles y elementos de diseño de interfaz de pantalla
* Para la pantalla principal decidimos colocar un fondo de oficina y para mostrar las acciones que se pueden realizar sobre la misma utilizamos 5 controles:

1. Puerta: es un dibujo representativo que se encuentra ubicado a la izquierda de la Pantalla Principal y al presionarlo genera el evento click que lo redirige a la Pantalla Inicial.
2. Mapa mundi: es un dibujo representativo que se encuentra ubicado a la izquierda de la Pantalla Principal y al presionarlo genera el evento click que lo redirige a la Pantalla Viaje.
3. Diario ó Laptop ó Celular: es un dibujo representativo que se encuentra ubicado a la derecha de la Pantalla Principal y al presionarlo genera el evento click que lo redirige a la Pantalla Personajes.
4. Estante: es un dibujo representativo que se encuentra ubicado a la izquierda de la Pantalla Principal y al presionarlo genera el evento click que lo redirige a la Pantalla Formulario.
   1. Forma de navegación de interfaz de pantalla

* La navegación entre las pantallas será utilizando el touch por intermedio de botones u objetos representativos.

1. Pantalla Inicial –>> Pantalla Principal

Utilizando el botón “Jugar”.

1. Pantalla Inicial –>> Pantalla Menú Windows Phone 7

Utilizando el botón “Salir”.

1. Pantalla Principal ->> Pantalla Viaje

Presionando con el touch sobre el mapamundi

1. Pantalla Principal ->> Pantalla Personajes

Presionando con el touch sobre el diario ó laptop ó celular, dependiendo de la iteración.

1. Pantalla Principal ->> Pantalla Formulario

Presionando con el touch sobre el estante.

1. Pantalla Principal ->> Pantalla Inicial

Presionando con el touch sobre la puerta.

1. Pantalla Formulario ->> Pantalla Sospechosos

Utilizando el botón “Filtrar”.

* Los botones predefinidos del teléfono también permiten navegación entre pantallas. Cuando se desee regresar a la pantalla anterior se podrá hacer presionando el botón “Back”. Idénticamente si lo que se desea es abrir el navegador de forma de buscar en la web información acerca del nuevo destino en el que se encuentra el sospechoso, se podrá utilizar el botón “Search” que provee el propio teléfono.
  1. Formatos de impresión

No fueron definidos todavía.

1. Diseño de Datos

En esta sección se define la estructura de datos que utilizará el sistema, a partir de los requisitos funcionales y no funcionales establecidos para el sistema y las particularidades del entorno tecnológico, que consiga una mayor eficiencia en el tratamiento de los datos.

* 1. Especificación de la Distribución de Datos

Los datos de nuestra aplicación estarán distribuidos en los nodos Windows Phone 7 y en el servidor Azure.

En el Windows Phone 7, estarán las imágenes de las ciudades y datos menores. Los demás datos serán traídos desde el servidor para el juego que se esté ejecutando en el celular.

En Azure, habrá una base de datos relacional (SQL Azure Database), en la que se guardarán los datos relacionados a los usuarios, a los juegos que están jugando cada usuario, a las ciudades, de los famosos, de las noticias de ciudades y famosos para poder generar las pistas y los demás datos de juego. Físicamente, no se sabe bien donde estará el servidor que contiene a esa base, ya que la misma está situada en la “nube”, o sea que estarán en algún centro de datos de Microsoft, en alguna máquina en particular.