Universidad Nacional de San Agustín

Facultad de Producción y Servicios

Escuela Profesional de Ciencia de la Computación

Curso:

Ingeniería de Software

Docente:

Edgar Sarmiento

Alumnos:

Barrios Conejo, Selene

Cueva Flores, Jonathan Brandon

García Diaz, German Flavio

Herrera Cooper, Miguel Alexander

Montesinos Apaza, Sergio

AREQUIPA - PERÚ

2019

Contenido

1. Introducción

El documento tiene como finalidad definir las especificaciones funcionales, no funcionales y del sistema para la implementación de una aplicación de desarrollo cognitivo que permitirá a los usuarios mejorar sus habilidades matemática mediante la resolución de diferentes test con enfoque dinámico.

1.1. Propósito

El propósito principal es el aprendizaje dinámico de los niños en el rango de edades de 7 a 13 años los cuales a través de una metodología de aprendizaje de Recompensas ellos logran mejorar o aprender nuevos principios matemáticos.

1.2. Ámbito del Sistema

- Nombre Provisional **EduKids**
- El sistema es un juego el cual permite el desarrollo cognitivo y no es un sistema de chat
- Desarrolla habilidades matemáticas, con el objetivo de mejorar el tiempo de resolución de problemas y como meta agregar más módulos de aprendizaje y plataformas de uso.

1.3. Definiciones, Acrónimos y Abreviaturas

- Data. - Información registrada por el usuario el cual es accesible solo por el sistema.
- RECORD. - Puntuación máxima alcanzada.
- GUI. - Interfaz Gráfica de Usuario.
- RAID. - Un grupo/matriz redundante de discos independientes.
- Algoritmos Simétricos. - Criptografía basada en una sola clave pública.

2. DESCRIPCIÓN GENERAL

2.1. Perspectiva del Producto

Si bien cuando somos pequeños, comenzamos a sumar sin darnos cuenta utilizando diferentes objetos cotidianos (por ejemplo, si tengo dos manzanas y compro tres, ¿cuántas tengo?) con el paso de los años y el estudio de las matemáticas en el colegio, **algunos de nosotros comenzamos a sentir una antipatía hacia las matemáticas que ya no abandonaremos**.

EduKids busca mejorar tu habilidad para las matemáticas a través de una plataforma de juegos divertida y gratuita. La perspectiva de nuestro perfil, no indicaría que tan bien nos fue en los niveles propuestos en el juego, ya que nuestro avatar tendrá un aspecto más personalizado gracias a nuestros logros.

La interfaz de EduKids ofrece al jugador la opción como:

- Ver sus logros en el juego.
- Comprar ítems para su avatar mediante una tienda.
- Obtener ayudas y/o mini tutoriales para resolver cualquier tipo de dudas que tenga acerca de los problemas propuestos.

El jugador puede seleccionar el sexo de su avatar, el avatar tiene objetos BASICOS y MEJORAS, los básicos ya están definidos por el sistema de acuerdo al sexo y las mejoras son las personalizaciones que se realizan después de haber obtenido monedas en el progreso del juego.

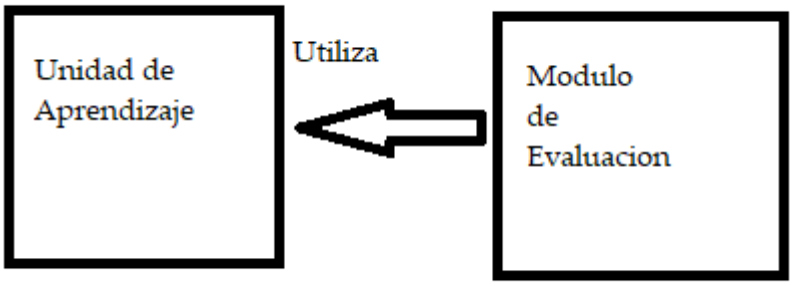
BASICOS:

MEJORAS:

Cuando empezamos una partida en EduKids , tenemos la fase de principiante (ELEGIMOS EL AVATAR BASICO) , para poder tener éxito en el juego debe lograr las metas propuestas de cada nivel.
El jugador en EduKids adquiere mayor experiencia en el juego cuando cumple las metas propuestas en menos tiempo, pero además recibe una recompensa especial.

2.2. Funciones del Producto

La aplicación EduKids , soporta el registro de usuarios, cada usuario tendrá un avatar que podrá modificar en su cuenta de perfil, también tendrá una tienda donde el usuario podrá comprar ítems para personalizar su avatar.
Edukids contiene unidades de aprendizaje, el usuario tendrá la capacidad de elegir una de estas unidades para que sea cargada por la aplicación (módulo de evaluación).
El módulo de evaluación analizara las respuestas del usuario para calcular el número de aciertos y errores.
En base a los resultados entregados por el módulo de evaluación generara las recompensas ganadas por el usuario estas serán (monedas y estrellas).
Edukids guardara los datos del usuario en el servidor de la aplicación. cada vez que el usuario inicie sesión se cargara del servidor los datos relacionados.



2.3. Características de los Usuarios

Educación Primaria

- El usuario busca aprender y desarrollar nuevos conocimientos en operaciones matemáticas.
- Busca diversión en el aprendizaje continuo.

Educación Secundaria

- El usuario con una necesidad de mejorar la eficiencia en el cálculo de operaciones matemáticas.
- Busca diversión en una actividad de la cual es su agrado.

2.4. Restricciones

- Compatible con ordenadores que soporte Windows XP a superior.
- Funcional solo en plataformas Windows.
- El uso de un agente tercero en caso de que ocurra un conflicto interno.
- El usuario no tiene acceso a información de otros usuarios.
- El intercambio de información es encriptado de manera simétrica.
- La seguridad de paquetes entregados a través de internet será encriptada asegurando así la integridad de estos mismos.

2.5. Suposiciones y Dependencias

- Si se agregara un módulo de chat podría afectar el objetivo principal del sistema que es aprender.
- Si el sistema operativo cambiaria se tendría que reconstruir todos los módulos en un sistema diferente el cual afectaría la integridad del mismo.

- Depende de un sistema de seguridad propio el cual funciona en cualquier plataforma asegurando la **data.**

2.6. Requisitos Futuros

- Implementar el módulo carrera para los usuarios.
- Implementar la aplicación en Android.

3. REQUISITOS ESPECIFICOS

3.1. Requisitos Funcionales (RF)

Nro. RF	REQUISITO	Descripción
RF01	Registrar Usuario	El sistema permite el registro de un usuario que desee aprender matemática por medio de EduKids .
RF02	Ingresar al Sistema	El usuario puede acceder al juego después de la validación de su juego.
RF03	Visualizar perfil y avatar de usuario.	El sistema muestra la información del usuario, logros, niveles y su respectivo avatar
RF04	Administrar Niveles de Competencia.	El sistema genera una variación de la dificultad al percibir el progreso del usuario y asimismo entrega las recompensas de acuerdo a su nivel.
RF05	Realizar autoevaluación	El sistema permite la regresión a ejercicios ya antes hechos.
RF06	Obtener recompensas	El sistema genera una recompensa por cada logro completado.
RF07	Reproducir Tutoriales del Tema	El sistema ofrece un pequeño tutorial en caso de no poder resolver en cierta cantidad de intentos.
RF08	Personalizar y obtener accesorios del avatar.	El usuario puede personalizar su avatar por medio de la tienda, en la cual se intercambian los méritos obtenidas en los distintos niveles por accesorios para el avatar.
RF09	Mostrar soluciones de ejercicios	El usuario debe acceder a la resolución del ejercicio en el cual tiene dificultad para estar preparado para su evaluación final, la cual decide si cambia de nivel.
RF010	Recompensar logros	El usuario obtiene recompensas extras al realizar un RECORD en alguno de los niveles designados.
RF011	Recuperar contraseña	El sistema permite la recuperación de clave a través de preguntas de seguridad, sin dar a conocer la contraseña anterior.

Cuadro 1: RF Tabla 1

3.2. Requisitos No Funcionales (RNF)

Nro. RNF	TIPO	DESCRIPCION
RNF1	Eficiencia	El sistema es sostenible para una gran cantidad de usuarios por el principio de escalabilidad aplicado en este.
RNF2	Eficiencia	El sistema por su estructura genera respuestas rápidas a acciones del usuario.
RNF3	Seguridad	Los datos serán encriptados con algoritmos simétricos.
RNF4	Seguridad	Los datos no podrán ser accedido por ningún usuario ya que todo será cifrado.
RNF5	Usabilidad	Cuenta con un sistema de ayuda el cual estará destinado a usuarios con problemas típicos.
RNF6	Usabilidad	La GUI es visualmente entendible por cualquier tipo de usuario no importando su experiencia.
RNF7	Usabilidad	La GUI del sistema es amigable con el usuario.
RNF8	Usabilidad	En caso de error el sistema genera el mensaje respectivo.
RNF9	Fiabilidad	En caso de fallo del servidor los datos serán resguardados en copias de respaldo(RAID's). .

Cuadro 2: RNF tabla 2

APENDICE

MODULOS	REQUISITOS	ENCARGADO	ESTILOS
Implementar el registro de usuario	Registrar Usuario	• García Diaz, German Flavio	• Letterbox • Constructivist • Spreadsheet
Implementar la validación de Usuario	Ingresar al Sistema	• Cueva Flores, Jonathan Brandon	• Bulletin Board • Aspects
Implementar la ventana del perfil de Usuario	Visualizar perfil y avatar de usuario	• Barrios Cornejo, Selene	•
Implementar el módulo de Evaluación	Realizar Autoevaluación	• García Diaz, German Flavio	• Letterbox • Constructivist • Aspects

Recopilación de tutoriales y fuentes externas	Reproducir tutoriales del tema	• Barrios Cornejo, Selene	•
Diseño GUI	Personalizar y Obtener accesorios del avatar	• Montesinos Apaza, Sergio	• Complementos
Recompensas Especiales	Recompensar Logro	• Herrera Cooper, Miguel Alexander	•
Implementación de funcionalidad de recuperación de contraseña	Recuperar Contraseña	• Cueva Flores, Jonathan Brandon	• Bulletin Board • Aspects
Definir e implementar lo módulos de aprendizaje	Teórico	• Cueva Flores, Jonathan Brandon • Montesinos Apaza, Sergio	•
Definir los Niveles de dificultad en los módulos	Teórico	• Montesinos Apaza, Sergio • Cueva Flores, Jonathan Brandon	•

S.O.L.I.D.

Principio de Sustitución de Liskov (LSP)

Módulos: Implementar el módulo de Evaluación, Recompensas Especiales

Herencia en el proceso de generación de operaciones matemáticas aleatorias.

```cpp
#ifndef OPERATION_H
#define OPERATION_H
#include <bits/stdc++.h>
#include <dirent.h>

using namespace std;

class operation{
    map<int,vector<string>> A;
public:
    operation(){
        srand(time_t(nullptr));
    }
    string FuntionGenerator(int Level,int aditional=10,string abcd="+"){
        Level+=1;
        int a=rand() % (aditional*Level) + Level;
        int b=rand() % (aditional*Level) + Level;
        stringstream ss;
        ss << a <<abcd<<b;
        if(abcd=="+"){A[a+b].push_back(ss.str());}
        else if(abcd=="*"){A[a*b].push_back(ss.str());}
        else if(abcd=="-"){A[a-b].push_back(ss.str());}
        return ss.str();
    }
};

#endif // OPERATION_H

```

Principio de segregación de Interfaces (ISP)

Módulos: Implementar la validación de Usuario, Implementar el registro de usuario

GRASP

COHESION:

Cohesión Funcional: Cuando el módulo ejecuta una y sólo una tarea, teniendo un único objetivo a cumplir, se dice que tiene Cohesividad Funcional.

```
#include <functions.h>
#include <record.h>

using namespace std;

class system32{
    int score;
public:
    system32(){
        score=0;
    }
    bool login(string a,string b){
        if(isDir("USUARIOS//"+a)){
            ifstream VALOR;
            VALOR.open("USUARIOS//"+a+"//"+b+".txt");
            if(VALOR){
                return true;
            }
            return false;
        }
        return false;
    }
    void registre(string a, string b, string c, string d, string e){
        if(isDir("USUARIOS//"+a)==false){
            string aux="mkdir \"USUARIOS\"//"+a;
            system(aux.c_str());
            string aux1="USUARIOS/"+a+"//"+b+".txt";
            ofstream Indice(aux1,ios::binary|ios::app);
            record abc(a,b,c,d,e);
            Indice.write(reinterpret_cast<const char *>(&abc), sizeof(record));
            Indice.close();
            ofstream record("USUARIOS/"+a+"//"+"Score.txt");
            record<<0;
            record.close();
            cout<<"Complete";
        }
    }
};

#endif // SYSTEM_H

```

ACOPLAMIENTO:

Acoplamiento de Control: Cuando un módulo le envía a otro un elemento de control que determina la lógica de ejecución del mismo.

```cpp
#ifndef OPERATION_H
#define OPERATION_H
#include <bits/stdc++.h>
#include <dirent.h>

using namespace std;

class operation{
    map<int,vector<string>> A;
public:
    operation(){
        srand(time_t(nullptr));
    }
    string FuntionGenerator(int Level,int aditional=10,string abcd="+"){
        Level+=1;
        int a=rand() % (aditional*Level) + Level;
        int b=rand() % (aditional*Level) + Level;
        stringstream ss;
        ss << a <<abcd<<b;
        if(abcd=="+"){A[a+b].push_back(ss.str());}
        else if(abcd=="*"){A[a*b].push_back(ss.str());}
        else if(abcd=="-"){A[a-b].push_back(ss.str());}
        return ss.str();
    }
};

#endif // OPERATION_H

```

Principio de la Inversión de Dependencias (DIP)

Módulos: Implementar la ventana del perfil de Usuario, Implementar la validación de Usuario, Recompensas Especiales

- **GENERACION DE PERSONAJES YA QUE UNA SOLA IMAGEN ESTARA CONECTADA DIFERENTES USUARIOS (CODIGO AUN NO DISPONIBLE)**
- **INYECCION DE DEPENDENCIAS A UNA CLASE CONCRETA DE VALIDACION QUE GENERA COOKIE DE SESION (CODIGO AUN NO DISPONIBLE)**
- **Finalidad de reducir el acoplamiento en la Clase Recompensa, esto hará que los se reduzca la dependencia entre los módulos de código de dicha clase.**

D.D.D.

Aspectos Teóricos

Lenguaje Ubicuo:

Propuesta para crear un lenguaje común entre los programadores y usuarios; propone nombrar

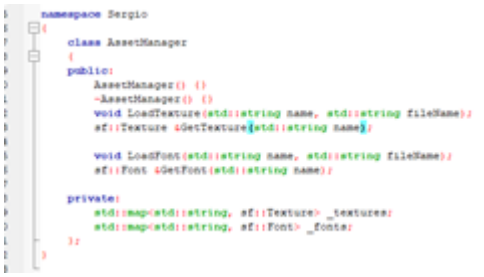
las variables, métodos y clases con lenguaje del dominio de modo que sea ‘auto explicable’, habitualmente los nombres representan objetos y los verbos representan métodos.

Capas de la Arquitectura

Se propone cuatro capas conceptuales:

A. User Interface

Presenta la información al usuario, interpreta sus acciones y las envía a la aplicación.



```
namespace Sergio
{
    class AssetManager
    {
    public:
        AssetManager() {}
        ~AssetManager() {}
        void LoadTexture(std::string name, std::string fileName);
        sf::Texture &GetTexture(std::string name);

        void LoadFont(std::string name, std::string fileName);
        sf::Font &GetFont(std::string name);

    private:
        std::map<std::string, sf::Texture> _textures;
        std::map<std::string, sf::Font> _fonts;
    };
}
```

```
#pragma once
#include "SFML\Graphics.hpp"

namespace Sergio
{
    class InputManager
    {
    public:
        InputManager() {}
        ~InputManager() {}

        bool IsSpriteClicked(sf::Sprite object, sf::Mouse::Button button, sf::RenderWindow &window);
        sf::Vector2i GetMousePosition(sf::RenderWindow &window);
    };
}
```

```
namespace Sergio
{
    class MainMenuState : public State
    {
    public:
        MainMenuState(GameDataRef data);
        void Init();
        void HandleInput();
        void Update(float dt);
        void Draw(float dt);

    private:
        void AddSprite(std::string title,sf::Vector2i pos);
        void AddSpriteInput(std::string title,sf::Vector2i pos);
        void AddText(std::string font,std::string title,sf::Vector2i pos,sf::Color color,short int Size);
        void AddTextDynamic(std::string font,std::string title,sf::Vector2i pos,sf::Color color,short int Size);
        GameDataRef _data;

        sf::Sprite _background;
        sf::Text _title;
        int sw;
        sf::String Input1;
        sf::Clock _clock;

        std::vector<sf::Sprite> _sprites;
        std::vector<sf::Sprite> _spritesInput;
        std::vector<sf::Text> _texts;
        std::vector<sf::Text> _texts_ag;

    };
```

B. Application

Encargada de verificar e interactuar con los diferentes usuarios durante el uso de la aplicación.

```cpp
#include <SFML/Graphics.hpp>
#include "StateMachine.hpp"
#include "AssetManager.hpp"
#include "InputManager.hpp"

namespace Sergio
{
    class GameData
    {   public:
        StateMachine machine;
        sf::RenderWindow window;
        AssetManager assets;
        InputManager input;
    };

    typedef std::shared_ptr<GameData> GameDataRef;

    class Game
    {
    public:
        Game(int width, int height, std::string title);

    private:
        // Updates run at 60 per second.
        const float dt = 1.0f / 60.0f;
        sf::Clock _clock;
        GameDataRef _data = std::make_shared<GameData>();
        void Run();
    };
}
```

C. Domain

Es el núcleo de la aplicación que contiene las reglas de negocio y el responsable de mantener el estado de los objetos de negocio.

En nuestro caso nuestro núcleo es el manejo de data y las operaciones matemáticas que la aplicación realiza.

D. Infrastructure

Capa de soporte para el resto de capas, provee la comunicación con las otras capas e implementa las persistencias de los objetos de negocio y las librerías de soporte para las otras capas.

Interconexión en ventanas de SFML y QT haciendo una aplicación con diferentes capas de aplicación.

__PARTE NO VISUAL AUN__(TEORICO)

E. Entities

Término conceptual cuya concreción a la hora de definir en nuestra aplicación que objetos y clases se tratarán como tales, dependerá del contexto y de los objetivos de este.

Nosotros generamos un registro en la Base de Datos para cada usuario.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
☐	1	**id**	varchar(20)	utf8_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	2	**name**	varchar(20)	utf8_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	3	**last_name**	varchar(20)	utf8_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	4	**password**	varchar(20)	utf8_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más
☐	5	**email**	varchar(100)	utf8_unicode_ci		No	*Ninguna*			Cambiar Eliminar Más

F. Values Objects

Conjunto de propiedades y comportamientos.

Almacenamiento de datos de los usuarios (características ya registradas)

id	name	last_name	password	email
test	test	test	test	tes@gmail.com

G. Services

Medio de relación de varios objetos, algunas claves para distinguirlos:

- Comportamientos que no pertenecen a ninguna entidad.
- Pueden intercambiarse con facilidad por servicios de similares características.
- Usados por varias entidades en varios puntos de la app.
- Nos interesa el resultado del proceso que el proceso en sí.
- No tienen un estado interno.

En nuestro caso al momento de realizar "Operaciones" la función que va a resolver esto no es la misma, es decir que una multiplicación no puede ser resuelta con restas.

H. Modules

Permiten seguir con facilidad el concepto de acoplamiento débiles y alta cohesión.

-Contamos con distintos módulos:

- Registro de usuario

- Interfaz de la app

- Verificación de Datos

- Realización de Operaciones

I. Aggregates

Grupos de entidades relacionadas entre sí a nivel de negocio. Cuando tenemos varias entidades relacionadas entre sí y son dependientes entre ellas, los agrupamos en 'Agregados'.

Al generar un inicio de sesión por parte del usuario este solo visualiza parte de sus informaciones el resto sigue estando oculto del resto de los usuarios.

J. Factories

Clases encargadas de crear entidades y agregados, construyendo todas las entidades contenidas en ellos.

Permiten abstraer y separar la lógica y reglas de creación de una entidad y dejar en las entidades únicamente las reglas de negocio que son inherentes a ellas.

- En nuestra aplicación aplicamos factoría en el momento que el usuario interactúa con la aplicación, ya sea en el momento que inicie sesión o realice operaciones.

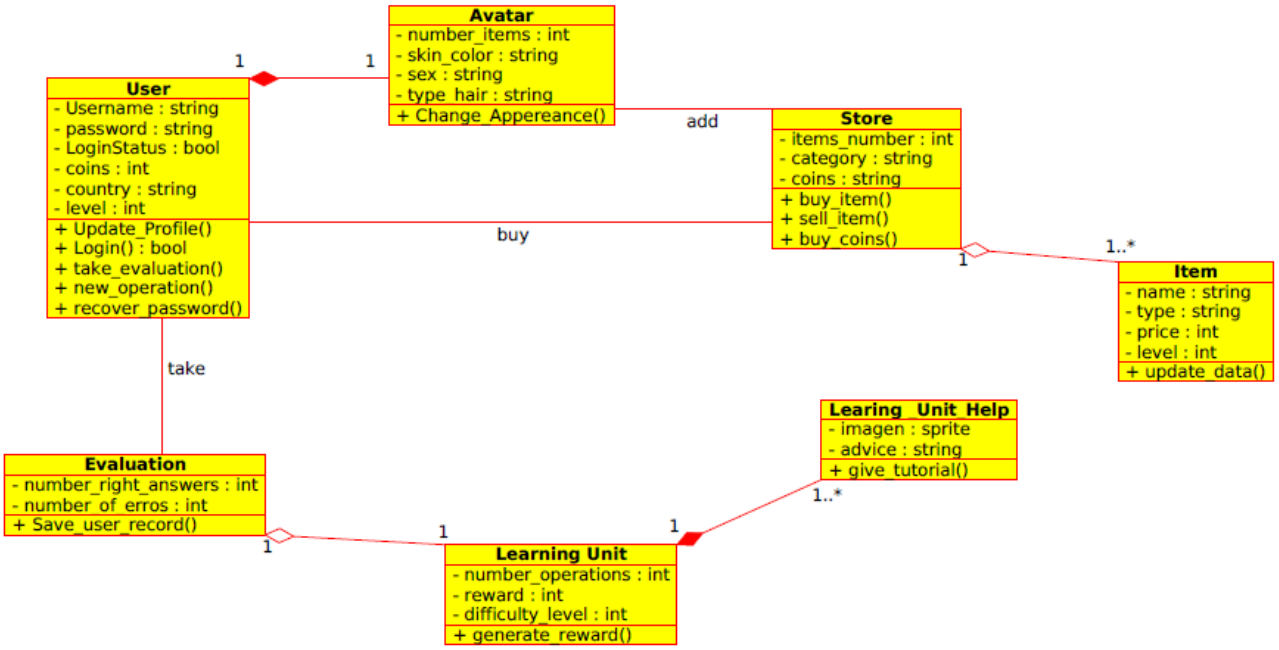
K. Repositories

La aplicación necesita mantener un puntero a cada objeto o entidad creada. Responsable de proporcionar a la aplicación a esos punteros.

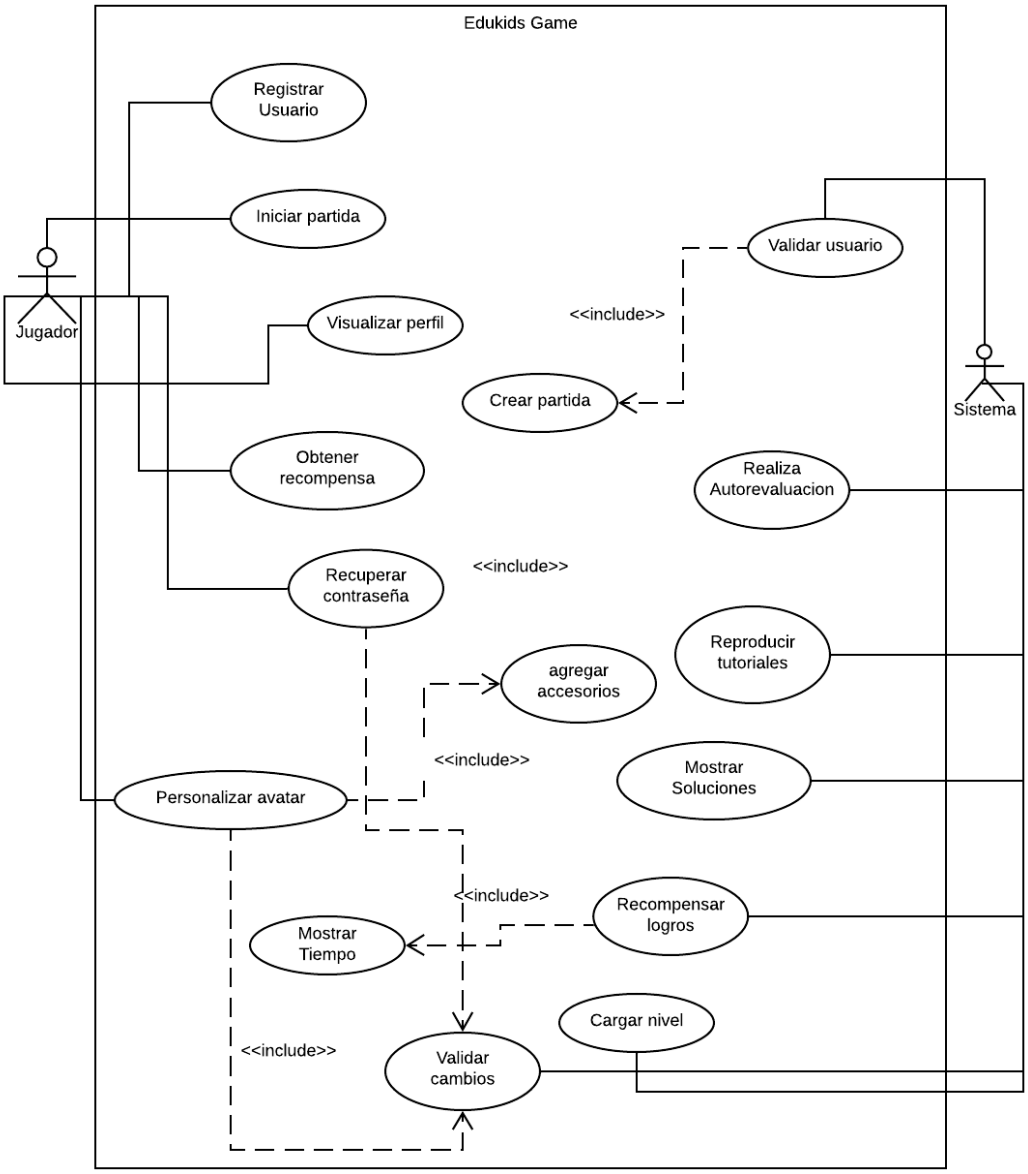
Conoce qué mecanismos se están utilizando para implementar la persistencia.

Contiene interfaces con métodos que permitan solicitar un puntero o instancia de una entidad, de varias utilizando filtro si es necesario y proporcionando a los métodos habituales de persistencia tales como inserción, modificación y borrado.

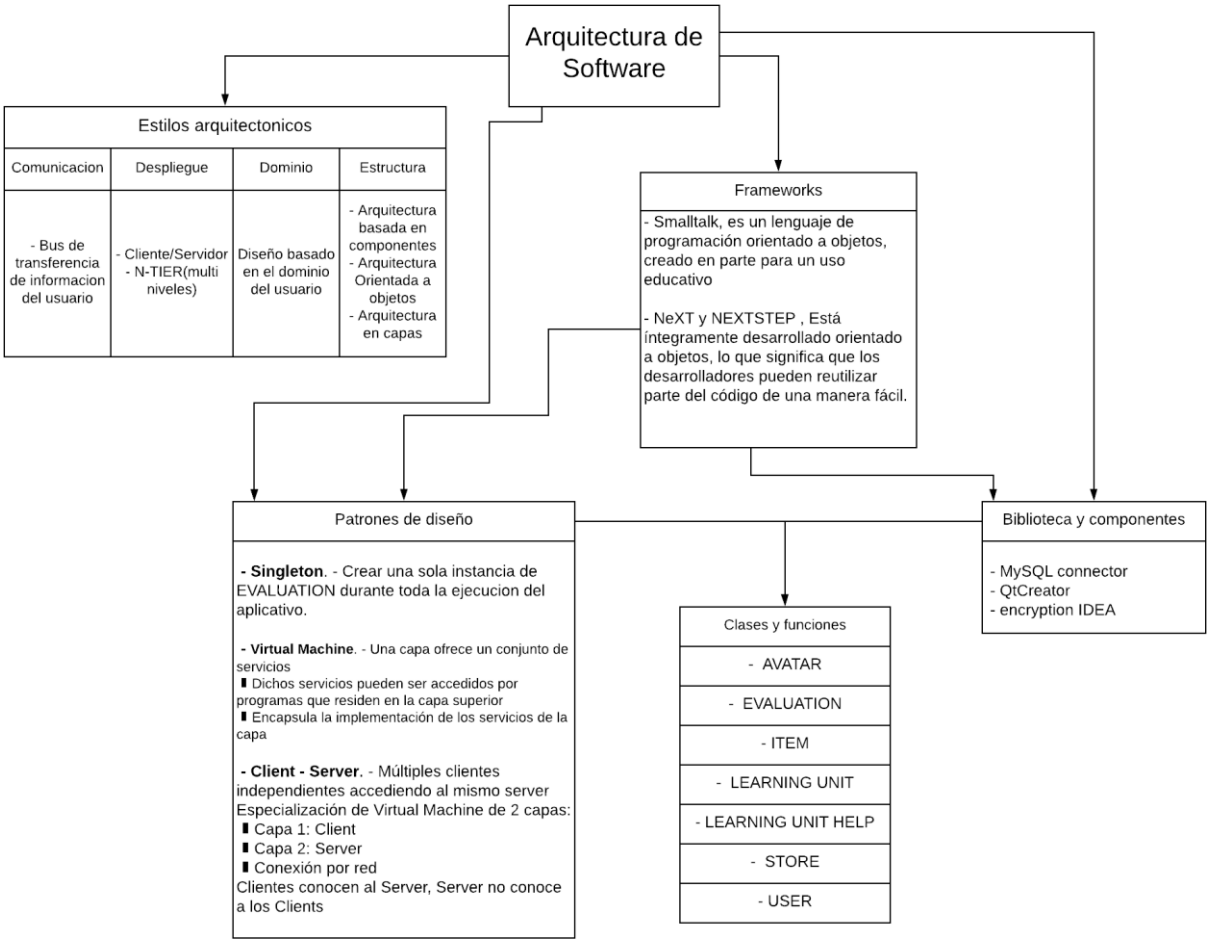
ANEXOS



Anexo 1: Diagrama de Clases



Anexo 2: Diagrama de Casos de Uso



Anexo 3: Arquitectura de Software