Actividad Final curso Fundamentos

Esta actividad es para realizar 100% individual, no deben existir actividades iguales en ninguna fase

Dado el siguiente escenario aplicar lo visto durante el curso.

1. Deben realizar el PB(product backlog)

2. Deben definir las actividades. su puntaje y su prioridad

3. Deben realizar el Sprint backlog

4. Definir el producto mínimo viable (MVP)

5. Deben realizar el diagrama de estados, secuencia, clases y entidad relación

6. Deben crear las tablas debidamente formalizadas (Normalizadas)

7. Crear las clases según el diagrama, implementando las buenas prácticas de programación

Yo como PO (Product owner) quiero un software que me permita reservar clases de un gimnasio, debe tener login, manejo de roles (usuario, coach, administrador)

debo poder como usuario registrar mi información así mismo como datos de contacto de emergencia y mis récords de peso.

Como coach debo poder ver quien agenda clase y ajustar el contenido de las clases que se dictan

Como administrador debo ver quien tiene activa la membresía y la información de los usuarios.

Roles de usuario:

Administrador

Coach

Usuario

**Product Backlog**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Escenarios | Ca/zación | ID\_PBI | PBI | Descripción | input | output | Acceptance Criteria |
| Usuarios y control de acceso | Papá | HU1 | Como producto owner requiero que se habilite un espacio para ingresar credenciales de accesos para login | La pagina de inicio de sesión debe incluir el nombre y logo del gym en la parte superior central | N/A | Pantalla con logo, nombre del gym mas espacio para ingreso de credenciales (login) | El sistema debe permitir el ingreso a la app bajo credenciales válidas |
| Usuario y control de acceso | Papá | HU2 | Como owner requiero que se active una interfaz en la cual se habiliten los accesos a las diferentes funcionalidades según rol (admin, coach y usuario) para un mejor control y usabilidad de la app | El sistema debe reconocer y activar interfaz según el rol por medio de las credenciales de acceso | Credenciales de acceso | Interfaz de usuario con funciones propias de cada rol | El sistema permite visualizar e interactuar con funcionalidades según el rol del usuario |
| Datos Generales | Papá | HU3 | Como administrador requiero visualizar la lista de usuarios con su respectiva información para tener una visión mas amplia del negocio | Visual de usuarios registrados en el aplicativo | valores | Lista de usuarios | El sistema lista usuarios registrados |
| Datos Generales | Hijo | HU3-1 | Como administrador requiero un ckeck box en la lista de usuarios con opción de:  \*usuarios activos  \*Todos los usuarios para ver quienes tienen membresía activa para una mejor trazabilidad | Al seleccionar La lista de usuario activos debe tener detallado el estado de la membresía de cada uno | Click sobre check box usuarios activos | Lista de usuarios con membrecía activa | El sistema debe permitir ver la lista de usuarios con membresía activa |
| Gestión de agendamientos | Papá | HU4 | Como coach requiero una lista de las clases que se van a realizar en el gym y el acceso a poder ajustar contenido de cada una de ella para una mejor trazabilidad. | El usuario coach tendrá la posibilidad de aditar el contenido de cada una de las clases impartidas | Click en nombre de la clase <editar> | Clase actualizada y disponibilidad | La app muestra una lista de las clases con opción de editarlas |
| Gestión de agendamientos | Hijo | HU4-1 | Como coach requiero visualizar lista de usuario inscritos Y ASOCIADOS a cada una de las clases en HU4 para tener claridad sobre el numero de usuarios que asistirán por cada rutina. | Al hacer click en el nombre de la clase debe aparecer un listado de usuarios inscritos para dicha clase. | Click en clases > nombre de clase | Lista de usuarios asociados a cada una de las clases | La app permite ver datos de usuarios agendados para cada clase |
| Registro usuarios | Papá | HU5 | Como usuario requiero poder registrar mis datos personales como:   * Nombre * Apellido * Teléfono * Edad * Peso (kg)   También datos de Contacto de emergencia como:   * Nombre * Teléfono * Parentesco   Para llenado de base de datos | En la interfaz de usuario debe tener un panel para registrar datos personales y de contacto de emergencia | Click en registrarse | Formulario de registro de datos personales | El sistema brindará un Formulario de registro de usuario |
| Registro de usuarios | Hijo | HU5-1 | Como usuario requiero poder actualizar mi peso de manera periódica para llevar un registro de mi progreso en el gym | En la interfaz de usuario se mostrará un espacio para añadir un nuevo registro de PESO del usuario cada x cantidad de tiempo, los pesos anteriores quedarán de igual forma visibles en la app | Usuario logueado como usuario > click en PESO > INGRESAR PESO> GUARDAR | Récord de peso del usuario desde el mas reciente al mas antiguo | El sistema permite añadir un nuevo peso y visualizar el historial de las otras medidas anteriores |
|  |  |  |  |  |  |  |  |

***Prioridad y Complejidad***

A continuación, en el sprint backlog se hace referencia al puntaje o grado de dificultad que en mi opinión personal tiene cada uno de los productos entregables, de igual forma la prioridad de estos en el negocio.

**Sprint Backlog**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Backlog | ID PB | Descripción | complejidad | prioridad | fecha | fecha entrega |
| Sprint 1 | Hu1 | Pantalla para login | 13 | Alta |  |  |
| Sprint 1 | HU2 | Interfaz según rol | 34 | Alta |  |  |
| Sprint 1 | HU3 | Lista de usuarios | 8 | Media |  |  |
| Sprint 1 | Hu5 | Registro de datos de usuarios | 8 | Alta |  |  |
| Sprint 2 | HU3-1 | Usuarios activos |  | Media |  |  |
| Sprint 2 | HU4 | Ajuste contenido de clases |  | Alta |  |  |
| … | …. | …. | …. | … | …. |  |

***Producto mínimo entregable (MVP)***

Teniendo en cuenta la prioridad del negocio y el puntaje o grado de complejidad de cada uno de las historias de usuario se agregó los siguientes casos para trabajar en el Sprint 1.

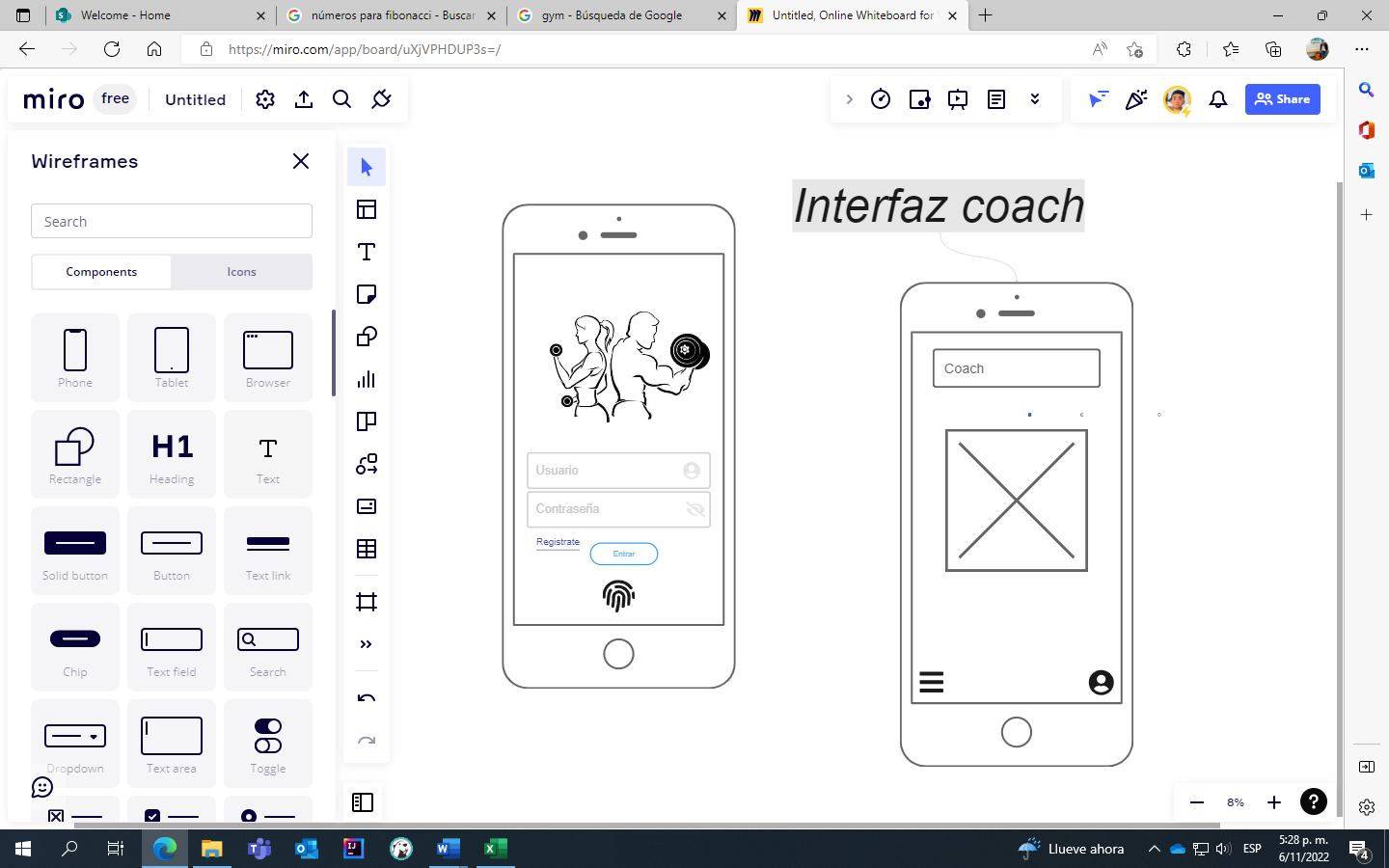
1. Un espacio para hacer login, en la cual los usuarios podrán ingresar sus credenciales para acceder al sistema el requerimiento se identifica en el ***product backlog como HU1***.

Interfaz de usuario gráfica

Descripción generada automáticamente

imagen silueta usada como logo sacado de [Pin en Top silhuetas (pinterest.es)](https://www.pinterest.es/pin/645281452848719546/)

1. Un espacio el cual tendrá funcionalidades dependiendo de cada rol de usuario y aparece identificado en el ***product backlog como HU2.***



Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Imagen Perfil coach Imagen Perfil Administrador

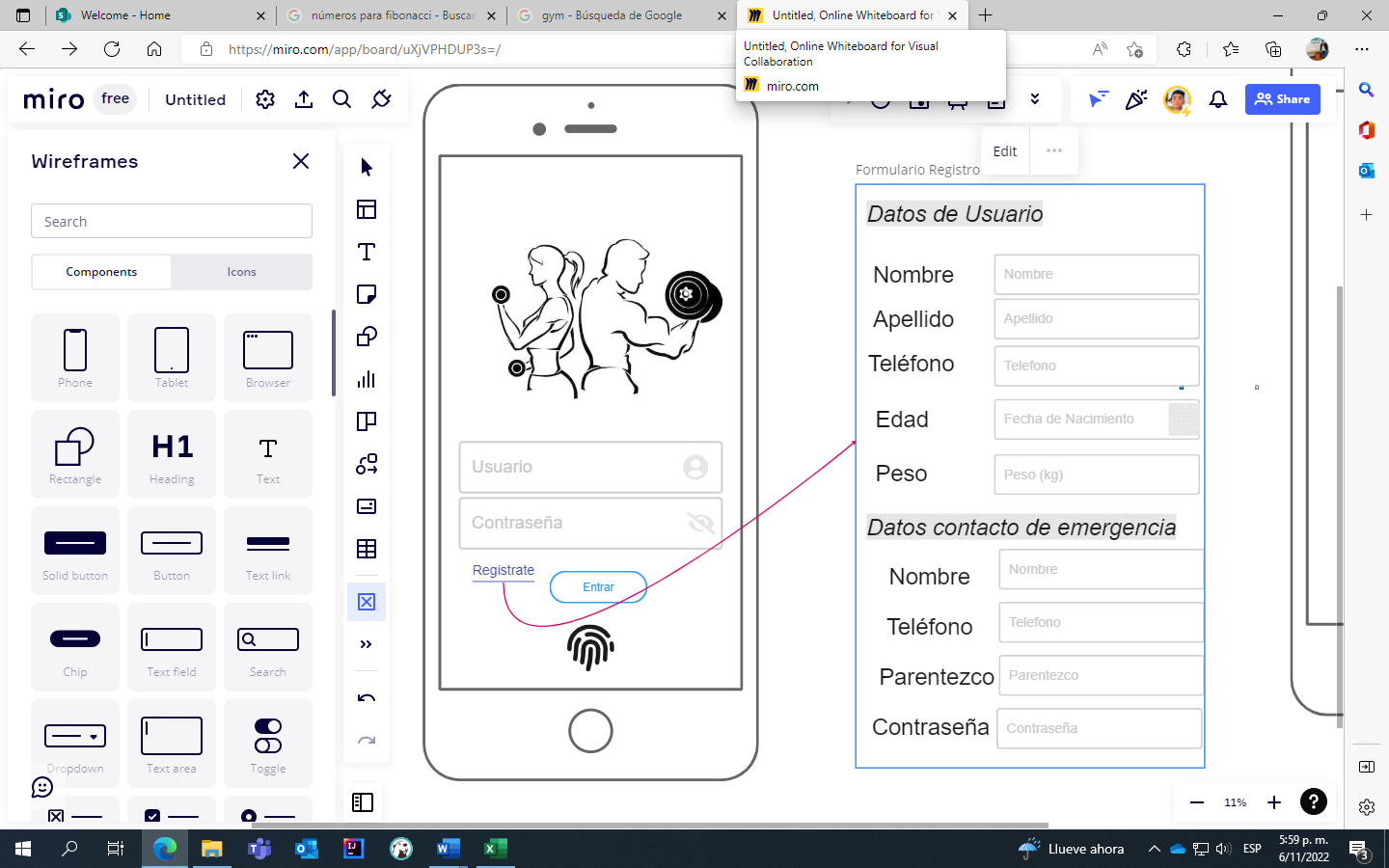
1. Un espacio donde se podrá consultar la lista de los usuarios registrados en el sistema y aparece identificado en el ***product backlog como HU3***.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

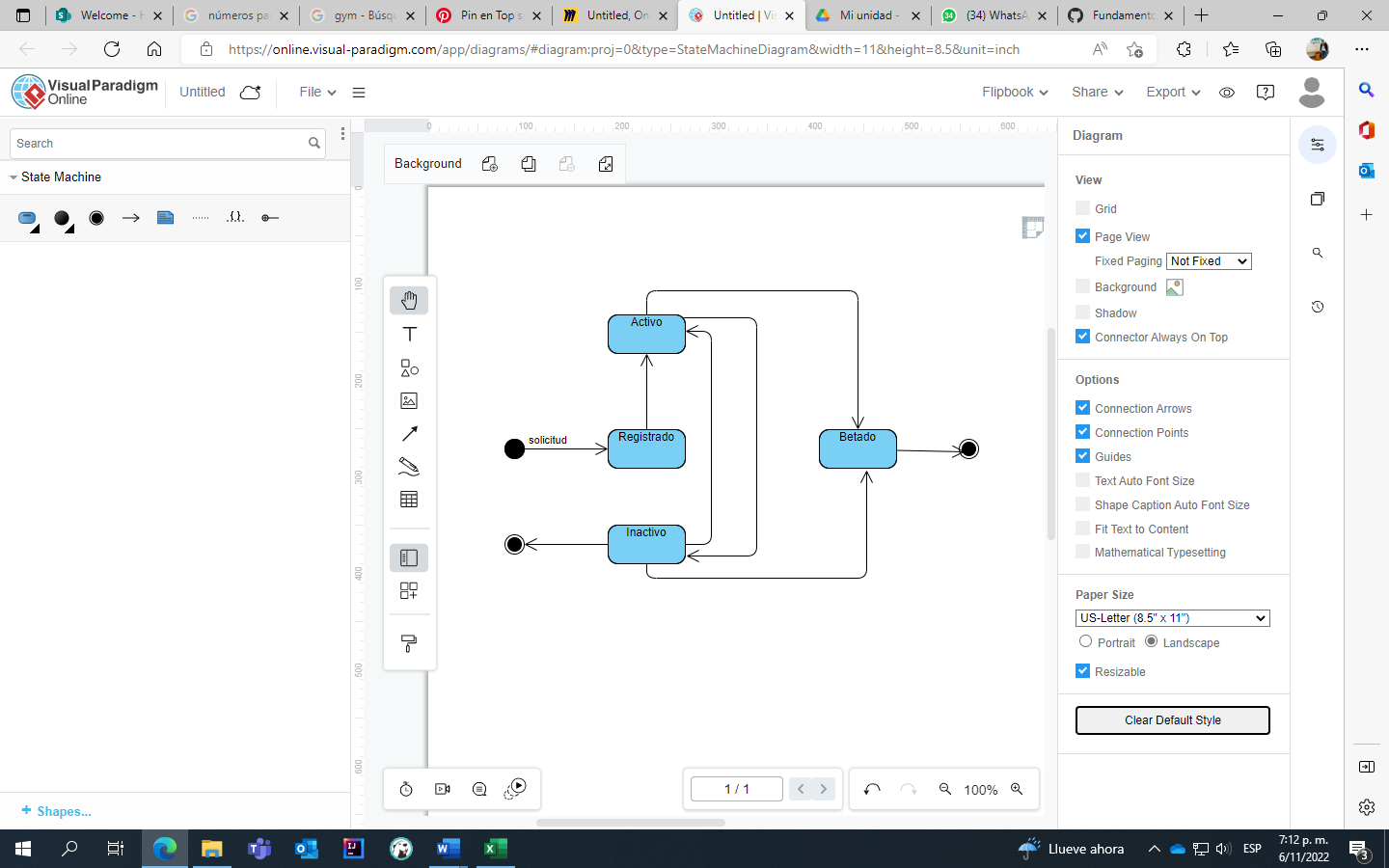
Imagen lista Usuarios registrados

1. Un espacio en el cual los usuarios puedan registrar sus datos personales el cual aparece identificado en el ***product backlog como HU5***

imagen silueta usada como logo sacado de [Pin en Top silhuetas (pinterest.es)](https://www.pinterest.es/pin/645281452848719546/)

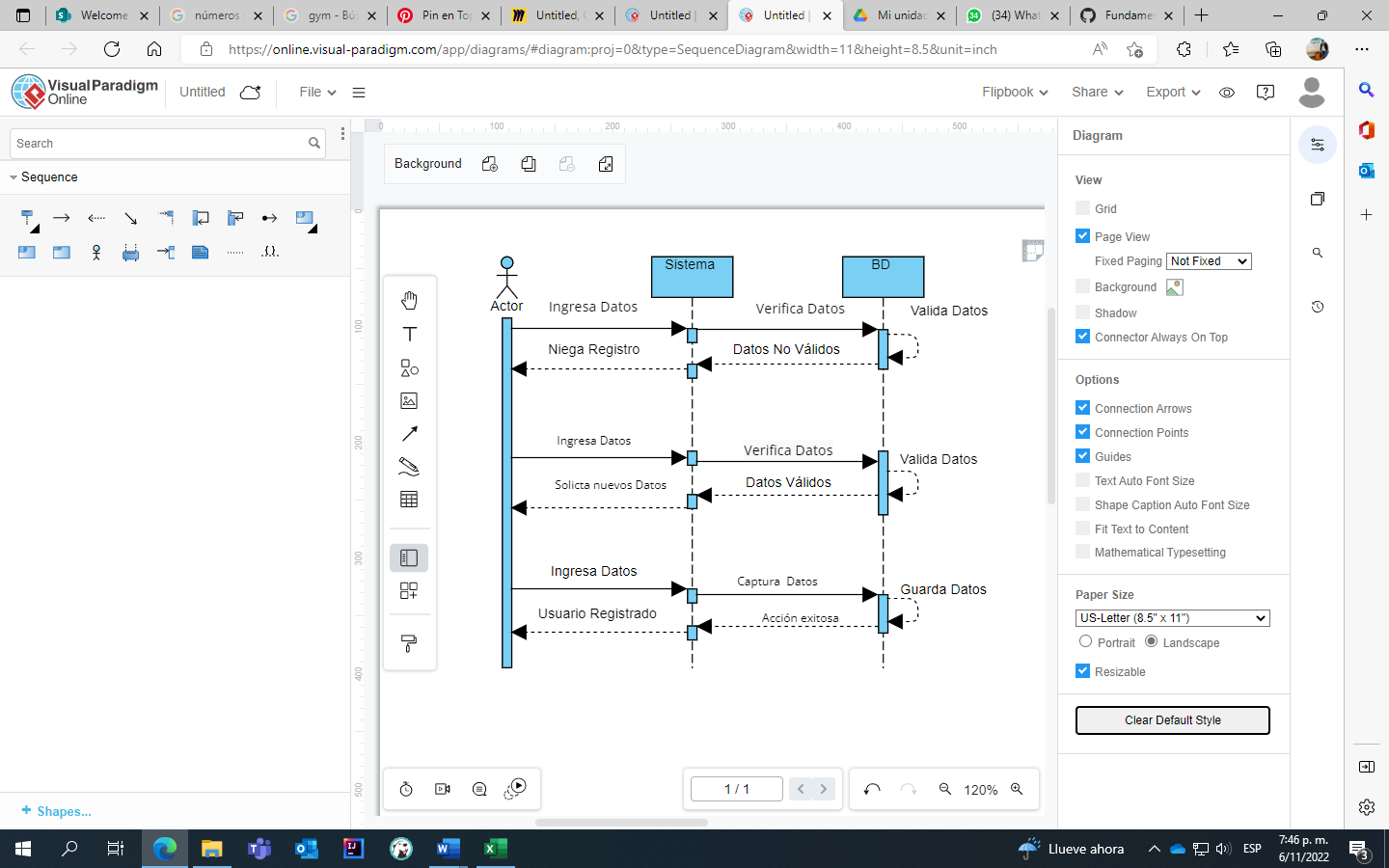
***Diagrama de Estado***

En el siguiente diagrama se plasma los distintos estados que puede tener una persona en el gym.

******

***Diagramas de Secuencia***

Muestra las interacciones de los objetos en determinado proceso, teniendo en cuenta el orden en que se producen



***Diagrama de Secuencia de Registro Usuario***

***Diagrama de Secuencia Buscar información de clientes***

El usuario debe estar logueado bajo el rol administrador

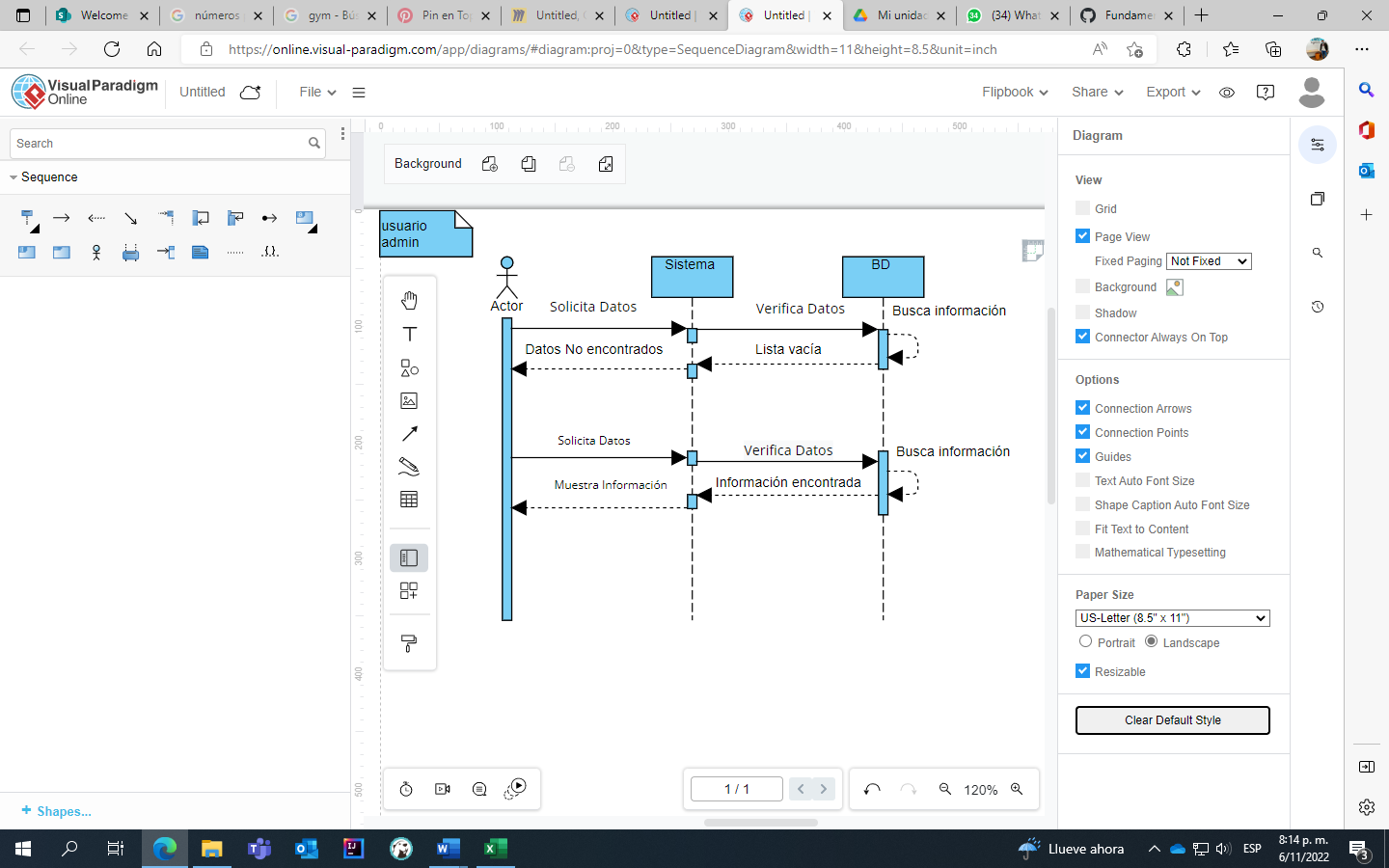
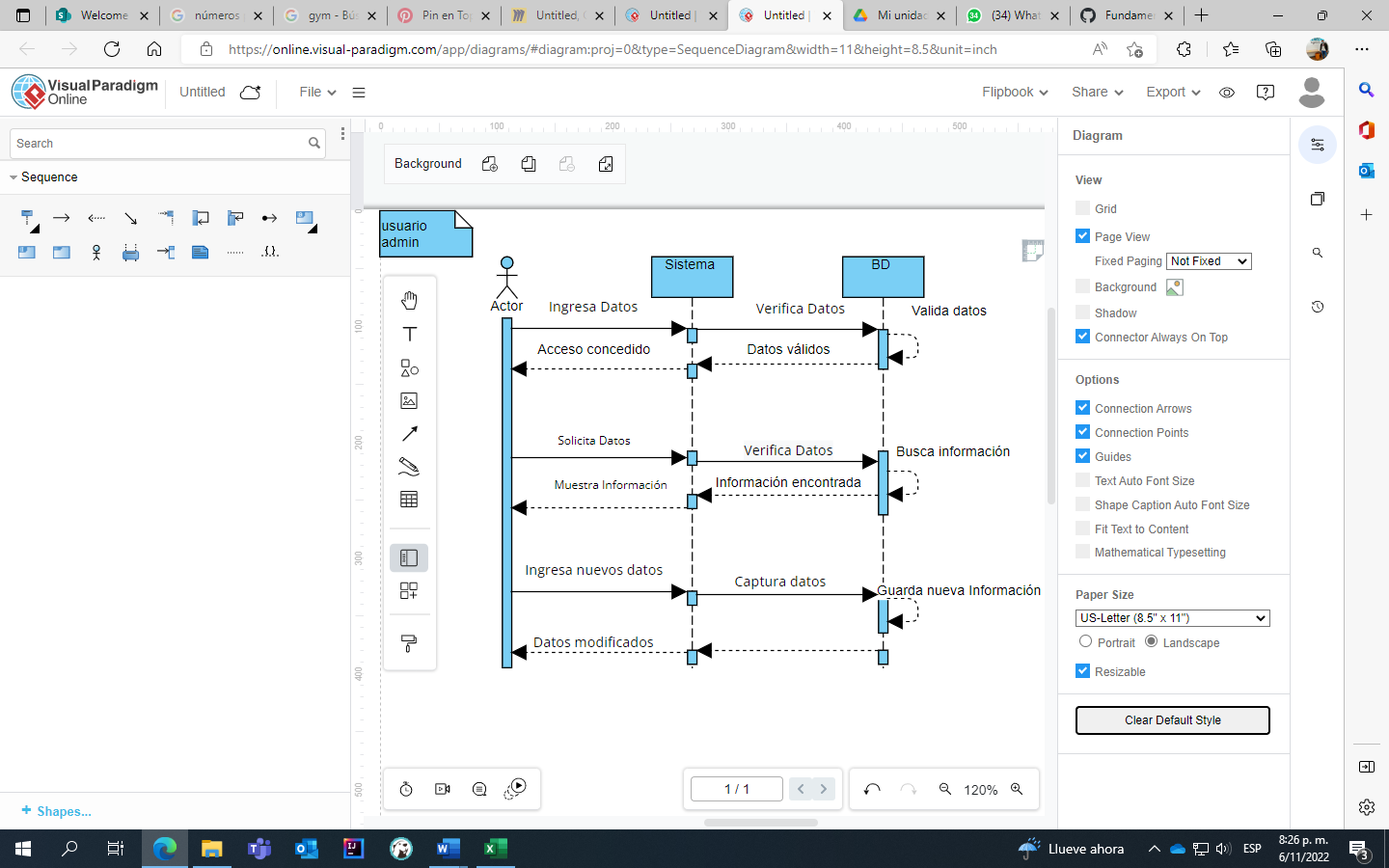


Diagrama secuencia mostrar información de usuarios

***Diagrama de Secuencia Ver usuarios agendados/Modificar contenido de Clases***

El usuario debe estar logueado bajo rol coach



***Diagrama de Clases***

Diagrama

Descripción generada automáticamenteen el diagrama de clases se trazan de forma clara la estructura del sistema, ya que se plasman sus clases con atributos y métodos con sus respectivas relaciones.

Imagen Diagrama de clases

***Diagrama

Descripción generada automáticamenteDiagrama entidad Relación***

***Código clase persona***

public class Persona {  
 private String documento;  
 private String nombre;  
 private String apellido;  
 private double peso;  
 private String fechaNacimiento;  
 private String telefono;  
private String contraseña;  
  
  
  
 //constructor  
 public Persona(String documento, String nombre, String apellido, double peso, String fechaNacimiento, String telefono, String contraseña) {  
 this.documento = documento;  
 this.nombre = nombre;  
 this.apellido = apellido;  
 this.peso=peso;  
 this.fechaNacimiento=fechaNacimiento;  
 this.telefono = telefono;  
 this.contraseña=contraseña;  
  
  
 }  
  
 public Persona() {  
  
 }  
  
 public String getDocumento() {  
 return documento;  
 }  
  
 public void setDocumento(String documento) {  
 this.documento = documento;  
 }  
  
 public String getNombre() {  
 return nombre;  
 }  
  
 public void setNombre(String nombre) {  
 this.nombre = nombre;  
 }  
  
 public String getApellido() {  
 return apellido;  
 }  
  
 public void setApellido(String apellido) {  
 this.apellido = apellido;  
 }  
  
 public double getPeso() {  
 return peso;  
 }  
  
 public void setPeso(double peso) {  
 this.peso = peso;  
 }  
  
 public String getFechaNacimiento() {  
 return fechaNacimiento;  
 }  
  
 public void setFechaNacimiento(String fechaNacimiento) {  
 this.fechaNacimiento = fechaNacimiento;  
 }  
  
 public String getTelefono() {  
 return telefono;  
 }  
  
 public void setTelefono(String telefono) {  
 this.telefono = telefono;  
 }  
  
 public String getContraseña() {  
 return contraseña;  
 }  
  
 public void setContraseña(String contraseña) {  
 this.contraseña = contraseña;  
 }

***Código clase Empleado***

public class Empleado extends Persona {  
  
 private int idEmpleado;  
 private String cargo;  
  
 //constructor  
 public Empleado(String documento, String nombre, String apellido, String telefono, String direccion) {  
 this.idEmpleado = idEmpleado;  
 this.cargo = cargo;  
  
  
 }  
  
 public int getIdEmpleado() {  
 return idEmpleado;  
 }  
  
 public void setIdEmpleado(int idEmpleado) {  
 this.idEmpleado = idEmpleado;  
 }  
  
 public String getCargo() {  
 return cargo;  
 }  
  
 public void setCargo(String cargo) {  
 this.cargo = cargo;  
 }

***código clase Administrador***

public class Administrador extends Persona {  
  
 private String usuario ="admin", password="125";  
  
 private int id\_admin;  
  
 public Administrador(String documento, String nombre, String apellido, double peso, String fechaNacimiento, String telefono, String contraseña, int id\_admin) {  
 super(documento, nombre, apellido, peso, fechaNacimiento, telefono, contraseña);  
 this.id\_admin = id\_admin;  
 this.usuario=usuario;  
 this.password=password;  
 }  
  
 public Administrador() {  
  
 }  
  
  
  
 public int getId\_admin() {  
 return id\_admin;  
 }  
  
 public void setId\_admin(int id\_admin) {  
 this.id\_admin = id\_admin;  
 }  
  
 public String getUsuario() {  
 return usuario;  
 }  
  
 public void setUsuario(String usuario) {  
 this.usuario = usuario;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
  
  
}

***código clase Usuario***

public class Usuario extends Persona {  
  
 private int idUsuario;  
  
 public Usuario(String documento, String nombre, String apellido, double peso, String fechaNacimiento, String telefono, String contraseña, int idUsuario) {  
 super(documento, nombre, apellido, peso, fechaNacimiento, telefono, contraseña);  
 this.idUsuario = idUsuario;  
 }  
  
 public int getIdUsuario() {  
 return idUsuario;  
 }  
  
 public void setIdUsuario(int idUsuario) {  
 this.idUsuario = idUsuario;  
 }  
}

***código clase Clase***

import java.util.Scanner;  
  
public class Clase {  
  
private int id\_clase;  
private String nombreClase;  
private String horario;  
  
 public Clase() {  
  
 }  
  
 public Clase(int id\_clase, String nombreClase, String horario) {  
 this.id\_clase = id\_clase;  
 this.nombreClase=nombreClase;  
 this.horario=horario;  
 }  
  
 public int getId\_clase() {  
 return id\_clase;  
 }  
  
 public void setId\_clase(int id\_clase) {  
 this.id\_clase = id\_clase;  
 }  
  
 public String getNombreClase() {  
 return nombreClase;  
 }  
  
 public void setNombreClase(String nombreClase) {  
 this.nombreClase = nombreClase;  
 }  
  
 public String getHorario() {  
 return horario;  
 }  
  
 public void setHorario(Date horario) {  
 horario = horario;  
 }

***código clase Membresía***

public class Membresia {  
 private String id\_membresia;  
 private String nombre\_membr;  
 private int precio;  
  
 public Membresia(String id\_membresia, String nombre\_membr, int precio) {  
 this.id\_membresia = id\_membresia;  
 this.nombre\_membr=nombre\_membr;  
 this.precio=precio;  
 }  
  
 public String getId\_membresia() {  
 return id\_membresia;  
 }  
  
 public void setId\_membresia(String id\_membresia) {  
 this.id\_membresia = id\_membresia;  
 }  
  
 public String getNombre\_membr() {  
 return nombre\_membr;  
 }  
  
 public void setNombre\_membr(String nombre\_membr) {  
 this.nombre\_membr = nombre\_membr;  
 }  
  
 public int getPrecio() {  
 return precio;  
 }  
  
 public void setPrecio(int precio) {  
 this.precio = precio;  
 }  
}

***código para ContactoEmergencia***

public class ContactoEmergencia {  
 String id\_usuario;  
 String nombre;  
 String apellido;  
 String telefono;  
 String parentezco;  
  
 public ContactoEmergencia(String id\_usuario,String nombre,String apellido,String telefono, String parentezco) {  
 this.id\_usuario = id\_usuario;  
 this.nombre=nombre;  
 this.apellido=apellido;  
 this.telefono=telefono;  
 this.parentezco=parentezco;  
 }  
  
 public String getId\_usuario() {  
 return id\_usuario;  
 }  
  
 public void setId\_usuario(String id\_usuario) {  
 this.id\_usuario = id\_usuario;  
 }  
  
 public String getNombre() {  
 return nombre;  
 }  
  
 public void setNombre(String nombre) {  
 this.nombre = nombre;  
 }  
  
 public String getApellido() {  
 return apellido;  
 }  
  
 public void setApellido(String apellido) {  
 this.apellido = apellido;  
 }  
  
 public String getTelefono() {  
 return telefono;  
 }  
  
 public void setTelefono(String telefono) {  
 this.telefono = telefono;  
 }  
  
 public String getParentezco() {  
 return parentezco;  
 }  
  
 public void setParentezco(String parentezco) {  
 this.parentezco = parentezco;  
 }  
}

***código clase Main***

import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.Scanner;  
  
  
public class Main {  
 public static void main(String[] args) {  
 List<Persona> personas = new ArrayList<>();  
  
 Scanner sc = new Scanner(System.*in*);  
 String documento, nombre, apellido, telefono, contraseña;  
 double peso;  
 String fechaNacimiento;  
 int opcion = 0;  
  
  
 while (true) {  
  
 System.*out*.println(" ===================================");  
 System.*out*.println(" Inicia Sesión ");  
 System.*out*.println("------------------------------------- ");  
 System.*out*.println(" 1. Regístrate ");  
 System.*out*.println(" 2. Iniciar Sesión ");  
 // System.out.println(" 3. Consultar Rutinas ");  
 System.*out*.println(" 4. Cambiar Contraseña ");  
 System.*out*.println(" 5. Cerrar Sesión ");  
 System.*out*.println(" ===================================");  
 System.*out*.print(" Que opción desea realizar: ");  
 //System.out.print(" ============================");  
 opcion = sc.nextInt();  
  
  
 switch (opcion) {  
 case 1:  
  
 Persona registro = new Persona();  
 System.*out*.println(" ===================================");  
 System.*out*.println(" - --Digite datos del usurario --- ");  
 System.*out*.print(" Documento: ");  
 documento = sc.next();  
 registro.setDocumento(documento);  
  
 System.*out*.print(" Nombre: ");  
 nombre = sc.next();  
 registro.setNombre(nombre);  
  
 System.*out*.print(" Apellido: ");  
 apellido = sc.next();  
 registro.setApellido(apellido);  
  
 System.*out*.print(" Peso: ");  
 peso = sc.nextDouble();  
 registro.setPeso(peso);  
  
 System.*out*.println(" Fecha de Nacimiento: ");  
 fechaNacimiento = sc.next();  
 registro.setFechaNacimiento(fechaNacimiento);  
  
 System.*out*.print(" Telefono: ");  
 telefono = sc.next();  
 registro.setTelefono(telefono);  
  
 System.*out*.print(" Contraseña: ");  
 contraseña = sc.next();  
 registro.setContraseña(contraseña);  
  
 personas.add(registro);  
  
 break;  
 case 2:  
 System.*out*.println(" ===================================");  
 if (personas.size() != 0) {  
 System.*out*.println(" - Digite su Nombre y Contraseña -");  
 System.*out*.print("Nombre: ");  
 nombre = sc.next();  
 System.*out*.print("Contraseña: ");  
 contraseña = sc.next();  
  
 for (Persona registroPersona : personas) {  
 if (registroPersona.getNombre().equals(nombre) && registroPersona.getContraseña().equals(contraseña)) {  
 System.*out*.println("Hola " + nombre + "!\n Bienvenido Al Gym ");  
  
 }  
  
 ArrayList<Clase> listaclase = new ArrayList<>();  
 while (opcion != 2) {  
 System.*out*.println(" =================");  
 System.*out*.println(" consulta aqui las clases -----");  
 System.*out*.println("1. Agendar clase ");  
 System.*out*.println("2. Atrás ");  
 System.*out*.print("Escribe la opcion:");  
 opcion = sc.nextInt();  
 switch (opcion) {  
  
 case 1:  
 System.*out*.println(" ====");  
 System.*out*.println("----Agrega una clase ---");  
 Clase metodos = new Clase();  
  
  
 listaclase.add(Clase.agregarclases());  
 break;  
 case 2:  
 break;  
  
 }  
  
 }  
 }  
 }  
 case 5: System.*out*.println("adios");  
 System.*exit*(0);  
 }  
  
 }  
 }  
}

***scripts BD***

***Créate table Usuario*** (

Id\_usuario int (5) primary key not null, nombre char (20), apellido char (20), teléfono char(15), fecha\_nacimiento date, contraseña char (8));

INSERT INTO usuario (id\_usuario, nombre, apellido, teléfono, fecha\_nacimiento, contraseña ) VALUES (‘01’, ´juan´,’perez’,’258752’, ‘25/12/1985’);

***Créate table membresía*** (

Id\_memb int (5) primary key not null, nombre char (20), valor double (7), estado boolean);

INSERT INTO membresía (id\_memb, nombre, valor, estado) VALUES (‘01’, ´gold´,’22000’,’true’);

***Créate table Administrador*** (

Id\_admin int (5) primary key not null, nombre char (20), apelllido char (20), teléfono char (15));

INSERT INTO membresía (id\_admin, nombre, apellido, telefono) VALUES (‘01’, ´pedro´,’leon’,’23658’);

***Créate table coach*** (

Id\_coach int (5) primary key not null, nombre char (20), apelllido char (20), teléfono char (15), usuario char (20),contraseña (8));

INSERT INTO coach (id\_coach, nombre, apellido, telefono, usuario, contraseña) VALUES (‘01’, ´tomas´,’soto’,’236848’, ‘tomas’, ‘25’);

***Créate table clase*** (

Id\_clase int (3) primary key not null, nombre char (10)

INSERT INTO coach (id\_clase, nombre) VALUES (‘01’, ´zumba´);