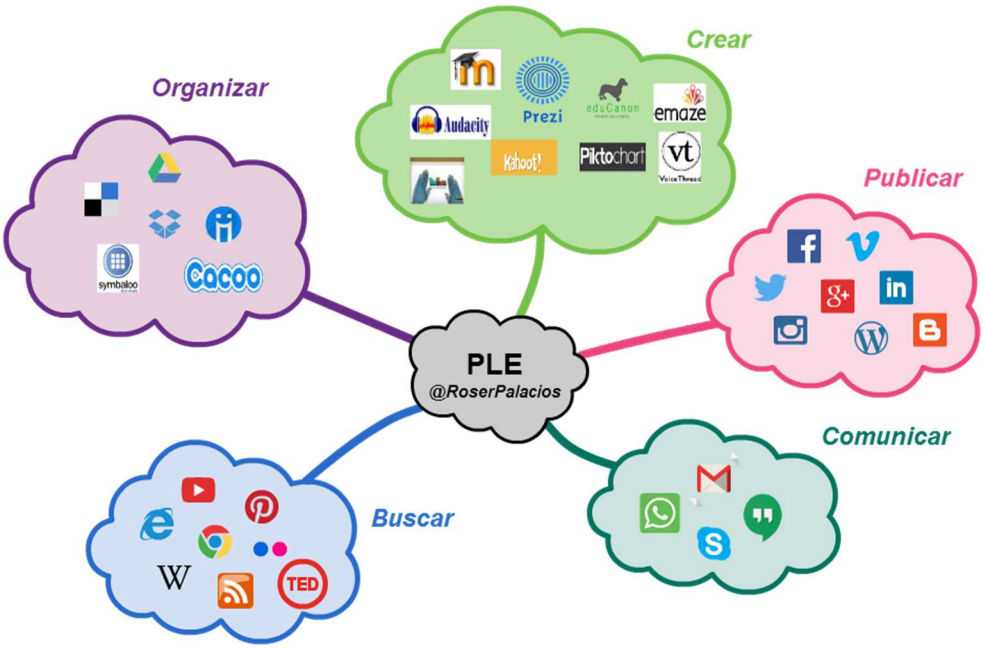
ENTORNOS DE DESARROLLO

PROYECTO PARA SUBIR NOTA



MIGUEL ANGEL DEL MORAL MARTINEZ | ENTORNOS DE DESARROLLO | 28 DE MAYO DE 2023

INDICE

1 INFORMACIÓN SOBRE LA PRÁCTICA

A lo largo del curso hemos visto todo tipo de diagramas, además de código en JAVA para poder realizar pruebas (última tarea).

En la tarea de Bases de Datos tenemos que crear la base de datos de un instituto y aquí vamos a documentarla.

Suponemos que vamos a crear una aplicación que va a gestionar dicha base de datos en JAVA (crear alumnos, consultarlos y modificarlos, lo mismo con asignaturas, profesores, notas etc.) todas las gestiones propias de una aplicación de gestión de institutos.

Necesitamos crear:

- Los diagramas de bases de datos
- Los diagramas de clases
- Los diagramas de comportamiento.
- Los diagramas de relación.
- Los casos de uso

¿Y para finalizar los casos de prueba de la aplicación, de que tipo? De al menos 2 de los que hemos visto.

Podemos crear alguna de las clases e implementar con JUNIT dichas pruebas por ejemplo.

Ahora vamos a ver, que hacemos con esta documentación 2 cosas:

Crear un PDF con toda la información y subirla a esta tarea del aula virtual.

Crear un repositorio de GIT y subir tanto la documentación como los ejemplos de código.

¿Qué es GIT? ¿Vamos a usar GIT o GITHUB?

Pues vamos a investigar un poco y vamos a poner dicha investigación en el documento, nos vamos a crear nuestro propio repositorio y a subir la dirección de nuestro repositorio con unas cuantas capturas al documento.

Tened en cuenta que este repositorio nos servirá de CV a la hora de buscar trabajo y demostrar todo cuanto sabemos hacer.

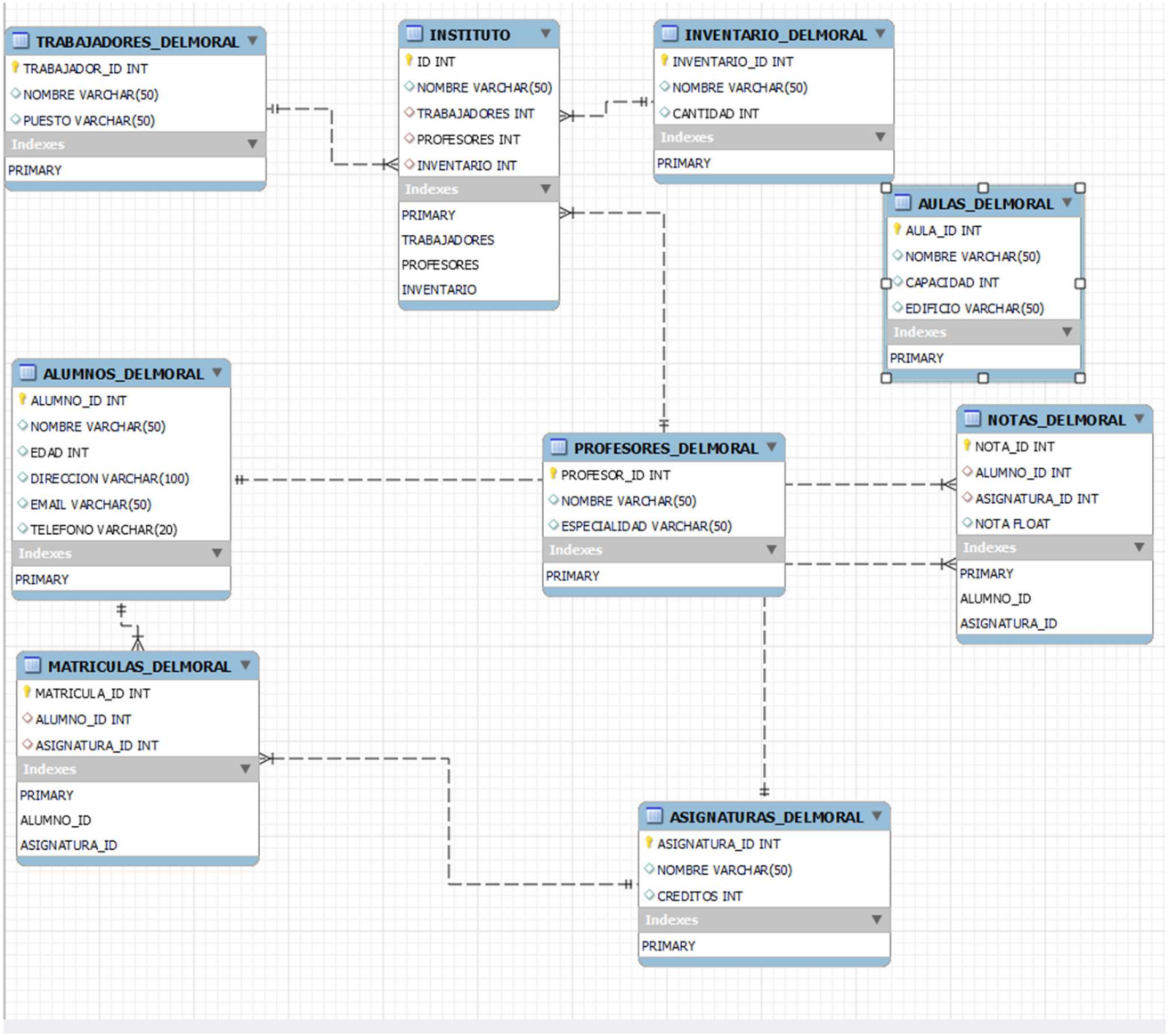
2 CREACIÓN DE LOS DIAGRAMAS DE BASE DE DATOS

El diagrama de bases de datos representa la estructura de la base de datos, incluyendo las tablas, columnas y relaciones entre ellas, para hacerlo algo más completos vamos a crear una tabla mas en la bb.dd con relaciones:

Las sentencias son:

```
-- CREAR TABLA INSTITUTO
CREATE TABLE INSTITUTO (
  ID INT PRIMARY KEY,
  NOMBRE VARCHAR(50),
  TRABAJADORES INT,
  PROFESORES INT,
  INVENTARIO INT,
  FOREIGN KEY (TRABAJADORES) REFERENCES TRABAJADORES_DELMORAL(TRABAJADOR_ID),
  FOREIGN KEY (PROFESORES) REFERENCES PROFESORES_DELMORAL(PROFESOR_ID),
  FOREIGN KEY (INVENTARIO) REFERENCES INVENTARIO_DELMORAL(INVENTARIO_ID)
);
```

El diagrama lo podemos generar automáticamente desde Workbench con la herramienta Reverse Enginer que nos daría como resultado lo siguiente



3 DIAGRAMAS DE CLASES

Un diagrama de clases es una representación visual de las clases, sus atributos y las relaciones entre ellas en un sistema. Cada clase representa un concepto o entidad en el sistema, y los atributos de la clase son las características o propiedades de esa entidad.

Clase "ALUMNOS_DELMORAL":

Atributos: ALUMNO_ID, NOMBRE, EDAD, DIRECCION, EMAIL y TELEFONO.

Métodos lógicos:

- getAlumnoID(): Retorna el ID del alumno.
- getNombre(): Retorna el nombre del alumno.
- getEdad(): Retorna la edad del alumno.
- getDireccion(): Retorna la dirección del alumno.
- getEmail(): Retorna el correo electrónico del alumno.
- getTelefono(): Retorna el número de teléfono del alumno.
- setEdad(edad): Actualiza la edad del alumno.
- setDireccion(direccion): Actualiza la dirección del alumno.

- setEmail(email): Actualiza el correo electrónico del alumno.
- setTelefono(telefono): Actualiza el número de teléfono del alumno.

Clase "ASIGNATURAS_DELMORAL":

Atributos: ASIGNATURA_ID, NOMBRE y CREDITOS.

Métodos lógicos:

- getAsignaturaID(): Retorna el ID de la asignatura.
- getNombre(): Retorna el nombre de la asignatura.
- getCreditos(): Retorna el número de créditos de la asignatura.
- setNombre(nombre): Actualiza el nombre de la asignatura.
- setCreditos(creditos): Actualiza el número de créditos de la asignatura.

Clase "MATRICULAS_DELMORAL":

Atributos: MATRICULA_ID, ALUMNO_ID y ASIGNATURA_ID.

Métodos lógicos:

- getMatriculaID(): Retorna el ID de la matrícula.
- getAlumnoID(): Retorna el ID del alumno asociado a la matrícula.
- getAsignaturaID(): Retorna el ID de la asignatura asociada a la matrícula.
- setAlumnoID(alumnoID): Actualiza el ID del alumno asociado a la matrícula.
- setAsignaturaID(asignaturaID): Actualiza el ID de la asignatura asociada a la matrícula.

Clase "PROFESORES_DELMORAL:

Atributos: PROFESOR_ID, NOMBRE y ESPECIALIDAD.

Métodos lógicos:

- getProfesorID(): Retorna el ID del profesor.
- getNombre(): Retorna el nombre del profesor.
- getEspecialidad(): Retorna la especialidad del profesor.
- setNombre(nombre): Actualiza el nombre del profesor.
- setEspecialidad(especialidad): Actualiza la especialidad del profesor.

Clase "INSTITUTO_DEL":

Atributos: ID, NOMBRE, TRABAJADORES, PROFESORES e INVENTARIO.

Métodos lógicos:

- getID(): Retorna el ID del instituto.
- getNombre(): Retorna el nombre del instituto.
- getTrabajadores(): Retorna la lista de trabajadores del instituto.
- getProfesores(): Retorna la lista de profesores del instituto.
- getInventario(): Retorna la lista de elementos en el inventario del instituto.
- setNombre(nombre): Actualiza el nombre del instituto.
- agregarTrabajador(trabajador): Agrega un trabajador a la lista de trabajadores del instituto.
- eliminarTrabajador(trabajador): Elimina un trabajador de la lista de trabajadores del instituto.
- agregarProfesor(profesor): Agrega un profesor a la lista de profesores del instituto.
- eliminarProfesor(profesor): Elimina un profesor de la lista de profesores del instituto.
- agregarElementoInventario(elemento): Agrega un elemento al inventario del instituto.
- eliminarElementoInventario(elemento): Elimina un elemento del inventario del instituto.
- obtenerCantidadTrabajadores(): Retorna el número de trabajadores en el instituto.

- obtenerCantidadProfesores(): Retorna el número de profesores en el instituto.
- obtenerCantidadElementosInventario(): Retorna la cantidad de elementos en el inventario del instituto.

El diagrama de clases representa la estructura estática de un sistema o aplicación, mostrando las clases, sus atributos y las relaciones entre ellas. En este caso, el diagrama muestra las clases "ALUMNOS_DELMORAL", "ASIGNATURAS_DELMORAL", "MATRICULAS_DELMORAL", "PROFESORES_DELMORAL" e "INSTITUTO_DEL". Las flechas en el diagrama indican las relaciones entre las clases, como la asociación entre "INSTITUTO_DELMORAL" y las otras clases.

El diagrama también muestra los atributos de cada clase, que representan las características o propiedades de los objetos de esa clase. Por ejemplo, la clase "ALUMNOS_DELMORAL" tiene atributos como "ALUMNO_ID" y "NOMBRE". Además de los atributos, los métodos lógicos se definen para cada clase, representando las operaciones o acciones que se pueden realizar con los objetos de esa clase.

En el caso de la clase "INSTITUTO_DELMORAL", los métodos lógicos incluyen la gestión de los trabajadores, profesores e inventario del instituto, permitiendo agregar, eliminar y obtener información relacionada con estos elementos.

En resumen, el diagrama de clases proporciona una representación visual de la estructura y relaciones de las clases en un sistema, mientras que los métodos lógicos definen las operaciones que se pueden realizar en cada clase para interactuar con los objetos y sus atributos.

La representación en tablas sería algo así

Clase "ALUMNOS_DELMORAL":
Atributos: ALUMNO_ID, NOMBRE, EDAD, DIRECCION, EMAIL y TELEFONO.
Métodos lógicos:
• getAlumnoID(): Retorna el ID del alumno. • getNombre(): Retorna el nombre del alumno. • getEdad(): Retorna la edad del alumno. • getDireccion(): Retorna la dirección del alumno. • getEmail(): Retorna el correo electrónico del alumno. • getTelefono(): Retorna el número de teléfono del alumno. • setEdad(edad): Actualiza la edad del alumno. • setDireccion(direccion): Actualiza la dirección del alumno. • setEmail(email): Actualiza el correo electrónico del alumno. • setTelefono(telefono): Actualiza el número de teléfono del alumno.

Clase "ASIGNATURAS_DELMORAL":
Atributos: ASIGNATURA_ID, NOMBRE y CREDITOS.
Métodos lógicos:
• getAsignaturaID(): Retorna el ID de la asignatura. • getNombre(): Retorna el nombre de la asignatura. • getCreditos(): Retorna el número de créditos de la asignatura. • setNombre(nombre): Actualiza el nombre de la asignatura. • setCreditos(creditos): Actualiza el número de créditos de la asignatura.

Clase "MATRICULAS_DELMORAL":
Atributos: MATRICULA_ID, ALUMNO_ID y ASIGNATURA_ID.
Métodos lógicos:
• getMatriculaID(): Retorna el ID de la matrícula.

- getAlumnoID(): Retorna el ID del alumno asociado a la matrícula.
- getAsignaturaID(): Retorna el ID de la asignatura asociada a la matrícula.
- setAlumnoID(alumnoID): Actualiza el ID del alumno asociado a la matrícula.
- setAsignaturaID(asignaturaID): Actualiza el ID de la asignatura asociada a la matrícula.

Clase "PROFESORES_DELMORAL:
Atributos: PROFESOR_ID, NOMBRE y ESPECIALIDAD.
Métodos lógicos:

- getProfesorID(): Retorna el ID del profesor.
- getNombre(): Retorna el nombre del profesor.
- getEspecialidad(): Retorna la especialidad del profesor.
- setNombre(nombre): Actualiza el nombre del profesor.
- setEspecialidad(especialidad): Actualiza la especialidad del profesor.

Clase "INSTITUTO_DELMORAL":
Atributos: ID, NOMBRE, TRABAJADORES, PROFESORES e INVENTARIO.
Métodos lógicos:

- getID(): Retorna el ID del instituto.
- getNombre(): Retorna el nombre del instituto.
- getTrabajadores(): Retorna la lista de trabajadores del instituto.
- getProfesores(): Retorna la lista de profesores del instituto.
- getInventario(): Retorna la lista de elementos en el inventario del instituto.
- setNombre(nombre): Actualiza el nombre del instituto.
- agregarTrabajador(trabajador): Agrega un trabajador a la lista de trabajadores del instituto.
- eliminarTrabajador(trabajador): Elimina un trabajador de la lista de trabajadores del instituto.
- agregarProfesor(profesor): Agrega un profesor a la lista de profesores del instituto.
- eliminarProfesor(profesor): Elimina un profesor de la lista de profesores del instituto.
- agregarElementoInventario(elemento): Agrega un elemento al inventario del instituto.
- eliminarElementoInventario(elemento): Elimina un elemento del inventario del instituto.
- obtenerCantidadTrabajadores(): Retorna el número de trabajadores en el instituto.
- obtenerCantidadProfesores(): Retorna el número de profesores en el instituto.
- obtenerCantidadElementosInventario(): Retorna la cantidad de elementos en el inventario del instituto.

4 DIAGRAMAS DE COMPORTAMIENTO.

Los diagramas de comportamiento son utilizados para representar el comportamiento dinámico de un sistema, mostrando cómo interactúan los diferentes elementos y cómo fluye la información entre ellos. A continuación, te muestro los diagramas de comportamiento más comunes:

Diagrama de secuencia: Este diagrama muestra la interacción entre objetos en una secuencia específica de eventos. Representa la secuencia de mensajes enviados entre los objetos y cómo se procesan esos mensajes en el tiempo. Los objetos se representan en líneas de vida verticales y los mensajes se muestran como flechas entre ellos.

Diagrama de actividad: Este diagrama representa el flujo de actividades y acciones dentro de un proceso. Muestra cómo se ejecutan las actividades y las decisiones que se toman en cada paso. Se utiliza para modelar el comportamiento detallado de un proceso, mostrando la secuencia de acciones y las condiciones que determinan el flujo.

Diagrama de estado: Este diagrama muestra los diferentes estados por los que pasa un objeto a lo largo de su vida y cómo cambia de un estado a otro en respuesta a eventos o condiciones. Se utiliza para representar el comportamiento dinámico de un objeto o de un sistema en términos de estados y transiciones.

Diagrama de comunicación: Similar al diagrama de secuencia, muestra la interacción entre objetos, pero se centra más en las asociaciones entre los objetos y las interacciones que ocurren en un escenario particular. Se utiliza para mostrar cómo los objetos se comunican entre sí y cómo se relacionan en un contexto específico.

Diagrama de colaboración: También conocido como diagrama de interacción, representa las interacciones entre los objetos y cómo se comunican y colaboran entre sí para lograr un objetivo común. Se centra en las relaciones entre los objetos y cómo se intercambian los mensajes.

Estos son solo algunos ejemplos de los diagramas de comportamiento más utilizados. Cada diagrama tiene su propia sintaxis y elementos específicos que se utilizan para representar el comportamiento del sistema.

Para realizarlas de forma automática existen varias herramientas de modelado UML que te permiten generar automáticamente diagramas a partir de código fuente o de descripciones textuales. Algunas de las herramientas populares son:

1. Enterprise Architect: Es una herramienta de modelado UML avanzada que permite generar automáticamente diagramas a partir de código fuente en varios lenguajes de programación. También cuenta con características de reversión de ingeniería para generar modelos UML a partir de sistemas existentes.
2. Visual Paradigm: Esta herramienta ofrece funcionalidades de generación automática de diagramas UML a partir de código fuente en varios lenguajes. También proporciona soporte para la generación de diagramas a partir de archivos de descripción textual, como archivos de texto plano o archivos XML.
3. Astah: Es una herramienta de modelado UML que permite generar automáticamente diagramas a partir de código fuente en varios lenguajes. También ofrece características de sincronización bidireccional, lo que significa que los cambios realizados en el código fuente se pueden reflejar automáticamente en los diagramas y viceversa.
4. Modelio: Esta herramienta de modelado UML tiene la capacidad de generar automáticamente diagramas a partir de código fuente en varios lenguajes de programación. También proporciona funciones de sincronización para mantener los diagramas y el código fuente actualizados.

Estas herramientas de modelado UML ofrecen una variedad de funcionalidades para la generación automática de diagramas, pero es importante tener en cuenta que la precisión y la calidad de los diagramas generados pueden variar según la complejidad del código fuente y la capacidad de la herramienta para analizarlo correctamente. Es recomendable probar diferentes herramientas y ajustar la configuración según tus necesidades específicas.

5 CASOS DE USO

A continuación, te describiré un ejemplo de casos de uso para la base de datos anterior:

Caso de uso: Crear alumno

Descripción: Permite crear un nuevo alumno en la base de datos.

Actores: Administrador, personal docente.

Flujo principal:

- El actor selecciona la opción de "Crear alumno" en la interfaz.
- El sistema muestra un formulario para ingresar los datos del nuevo alumno, como nombre, fecha de nacimiento, género, etc.
- El actor completa el formulario con los datos del alumno.

- El sistema valida los datos ingresados.
- El sistema guarda los datos del alumno en la tabla "ALUMNOS_DELMORAL" de la base de datos.

Caso de uso: Consultar alumno

Descripción: Permite buscar y visualizar los datos de un alumno en la base de datos.

Actores: Administrador, personal docente.

Flujo principal:

- El actor selecciona la opción de "Consultar alumno" en la interfaz.
- El sistema muestra una pantalla o formulario para ingresar el ID o algún dato de búsqueda del alumno.
- El actor proporciona los criterios de búsqueda.
- El sistema realiza una consulta en la tabla "ALUMNOS_DELMORAL" de la base de datos y muestra los resultados en la interfaz

Caso de uso: Modificar alumno

Descripción: Permite actualizar los datos de un alumno existente en la base de datos.

Actores: Administrador, personal docente.

Flujo principal:

- El actor selecciona la opción de "Modificar alumno" en la interfaz.
- El sistema muestra una pantalla o formulario para ingresar el ID o algún dato de búsqueda del alumno a modificar.
- El actor proporciona los criterios de búsqueda.
- El sistema realiza una consulta en la tabla "ALUMNOS_DELMORAL" de la base de datos para obtener los datos del alumno a modificar.
- El sistema muestra los datos del alumno en un formulario editable.
- El actor realiza las modificaciones necesarias en los datos del alumno.
- El sistema valida y actualiza los datos del alumno en la base de datos.

Caso de uso: Eliminar alumno

Descripción: Permite eliminar un alumno de la base de datos.

Actores: Administrador, personal docente.

Flujo principal:

- El actor selecciona la opción de "Eliminar alumno" en la interfaz.
- El sistema muestra una pantalla o formulario para ingresar el ID o algún dato de búsqueda del alumno a eliminar.
- El actor proporciona los criterios de búsqueda.
- El sistema realiza una consulta en la tabla "ALUMNOS_DELMORAL" de la base de datos para obtener los datos del alumno a eliminar.
- El sistema muestra los datos del alumno y solicita confirmación al actor.
- El actor confirma la eliminación.
- El sistema elimina los datos del alumno de la base de datos.

6 CASOS DE PRUEBA.

Caso de Prueba 1 - Creación exitosa de un alumno:

- Descripción: Verificar que se pueda crear correctamente un nuevo alumno con todos los datos requeridos.
- Entrada: Nombre = "Juan", Apellido = "Pérez", Fecha de Nacimiento = "1998-05-15".
- Proceso:
- Ejecutar la función "Crear Alumno" con los datos de entrada.
- Verificar que se haya creado un nuevo registro en la tabla "Alumnos" con los datos proporcionados.
- Salida Esperada: Se crea un nuevo alumno con los datos proporcionados en la tabla "Alumnos".

Caso de Prueba 2 - Error al crear un alumno sin nombre:

- Descripción: Verificar que se muestre un mensaje de error al intentar crear un alumno sin especificar el nombre.
- Entrada: Nombre = "", Apellido = "Gómez", Fecha de Nacimiento = "2000-10-20".
- Proceso:
- Ejecutar la función "Crear Alumno" con los datos de entrada.
- Verificar que se muestre un mensaje de error indicando que el nombre es obligatorio.
- Verificar que no se haya creado ningún nuevo registro en la tabla "Alumnos".
- Salida Esperada: Se muestra un mensaje de error indicando que el nombre es obligatorio y no se crea ningún registro nuevo en la tabla "Alumnos".

7 IMPLEMENTAR EN JUNIT

CASO DE PRUEBA 1

```
IMPORT ORG.JUNIT.TEST;
IMPORT STATIC ORG.JUNIT.ASSERT.*;

PUBLIC CLASS CREARALUMNOTEST {

    @TEST
    PUBLIC VOID TESTCREARALUMNOEXITOSO() {
        // DATOS DE ENTRADA
        STRING NOMBRE = "JUAN";
        STRING APELLIDO = "PÉREZ";
        STRING FECHANACIMIENTO = "1998-05-15";

        // LÓGICA PARA CREAR EL ALUMNO
        ALUMNO ALUMNO = CREARALUMNO(NOMBRE, APELLIDO, FECHANACIMIENTO);

        // VERIFICACIONES
        ASSERTNOTNULL(ALUMNO);
        ASSERTEQUALS(NOMBRE, ALUMNO.GETNOMBRE());
        ASSERTEQUALS(APELLIDO, ALUMNO.GETAPELLIDO());
        ASSERTEQUALS(FECHANACIMIENTO, ALUMNO.GETFECHANACIMIENTO());
    }

    // IMPLEMENTACIÓN DE LA FUNCIÓN "CREARALUMNO"
    PRIVATE ALUMNO CREARALUMNO(STRING NOMBRE, STRING APELLIDO, STRING FECHANACIMIENTO) {
        // LÓGICA PARA CREAR EL ALUMNO Y GUARDAR EN LA BASE DE DATOS
        // ...
        // RETORNAR EL OBJETO ALUMNO CREADO
    }
}
```

CASO DE PRUEBA 2

```
IMPORT ORG.JUNIT.TEST;
IMPORT STATIC ORG.JUNIT.ASSERT.*;

PUBLIC CLASS CREARALUMNOTEST {

    @TEST(EXPECTED = ILLEGALARGUMENTEXCEPTION.CLASS)
    PUBLIC VOID TESTCREARALUMNOSINNOMBRE() {
        // DATOS DE ENTRADA
        STRING NOMBRE = "";
        STRING APELLIDO = "GÓMEZ";
        STRING FECHANACIMIENTO = "2000-10-20";

        // LÓGICA PARA CREAR EL ALUMNO
        CREARALUMNO(NOMBRE, APELLIDO, FECHANACIMIENTO);

        // NO SE ALCANZARÁ ESTA LÍNEA DEBIDO A QUE SE ESPERA UNA EXCEPCIÓN
    }

    // IMPLEMENTACIÓN DE LA FUNCIÓN "CREARALUMNO"
    PRIVATE ALUMNO CREARALUMNO(STRING NOMBRE, STRING APELLIDO, STRING FECHANACIMIENTO) {
        IF (NOMBRE.ISEMPTY()) {
            THROW NEW ILLEGALARGUMENTEXCEPTION("EL NOMBRE ES OBLIGATORIO");
        }

        // LÓGICA PARA CREAR EL ALUMNO Y GUARDAR EN LA BASE DE DATOS
        // ...
        // RETORNAR EL OBJETO ALUMNO CREADO
    }
}
```

8 GIT Y GITHUB

GIT es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Permite realizar un seguimiento de los cambios en los archivos a lo largo del tiempo, colaborar con otros desarrolladores, fusionar cambios y revertir a versiones anteriores si es necesario. GIT proporciona un historial completo de todas las modificaciones realizadas en un proyecto y facilita la gestión de diferentes ramas de desarrollo.

Por otro lado, GitHub es una plataforma en línea basada en GIT que ofrece un servicio de alojamiento de repositorios de código. Permite a los desarrolladores almacenar y compartir sus proyectos, colaborar con otros usuarios y facilita la implementación continua (CI/CD) en un entorno colaborativo. GitHub ofrece características adicionales como seguimiento de problemas, gestión de proyectos y colaboración en equipo.

Para tu caso, puedes utilizar GIT para crear un repositorio local en tu máquina y luego utilizar GitHub para alojar ese repositorio en línea y compartirlo con otras personas. Puedes seguir los siguientes pasos:

Instala GIT en tu máquina siguiendo las instrucciones específicas para tu sistema operativo.

Crea un repositorio local utilizando el comando git init. Esto creará un nuevo repositorio GIT en la carpeta seleccionada.

Agrega tus archivos de documentación y ejemplos de código al repositorio utilizando el comando git add.

Realiza un commit de los cambios utilizando el comando git commit -m "Mensaje descriptivo". Esto registrará los cambios en el repositorio.

Crea un repositorio en GitHub y sigue las instrucciones para subir tu repositorio local a GitHub. Esto te proporcionará una dirección URL para tu repositorio en línea.

Agrega la dirección URL de tu repositorio de GitHub al documento junto con algunas capturas de pantalla para ilustrar la creación del repositorio y la subida de los archivos.

Recuerda que Git y GitHub son herramientas muy útiles para la gestión de versiones y la colaboración en proyectos de desarrollo de software. Tener un repositorio en línea con tus proyectos y ejemplos de código puede ser una excelente manera de mostrar tus habilidades y experiencia a posibles empleadores.

git --fast-version-control

1. The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

9 Download for Windows

2. **Click here to download** the latest (**2.40.1**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **about 1 month ago**, on 2023-04-25.

10 Other Git for Windows downloads

11 Standalone Installer

3. **32-bit Git for Windows Setup.**
4. **64-bit Git for Windows Setup.**

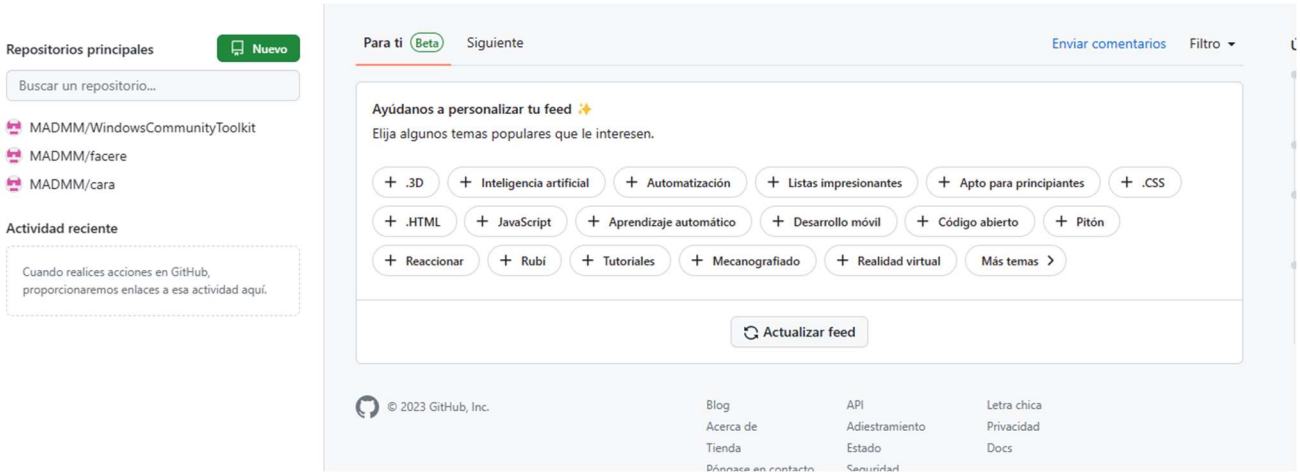
12 Portable ("thumbdrive edition")

5. **32-bit Git for Windows Portable.**
6. **64-bit Git for Windows Portable.**

Ahora vemos GitHub

GitHub

Aquí tengo varios repositorios



Desde este enlace se puede acceder a mi repositorio MADMM/Entornos_de_Desarrollo: Curso 1 DAM de entornos de desarrollo (github.com)