



Universidad
Nacional de
General
Sarmiento



TRABAJO PRÁCTICO LABORATORIO DE CONSTRUCCIÓN DE SOFTWARE

Segundo Cuatrimestre
2022



Licenciatura en Sistemas

Alumnos:

Enzo Arebalos
José Luis Quiñones

Profesores:

Ing. Juan Carlos Monteros
Ing. Francisco Orozco De La Hoz
Lic. Leandro Dikenstein

Índice

<i>Introducción</i>	<i>3</i>
<i>Objetivo</i>	<i>4</i>
<i>Configuración de entorno</i>	<i>5</i>
<i>Base de datos</i>	<i>7</i>
<i>Desarrollo</i>	<i>9</i>
<i>Interfaz de usuario</i>	<i>13</i>
<i>Reporte</i>	<i>15</i>
<i>Instalador.....</i>	<i>16</i>
<i>Dificultades</i>	<i>17</i>
<i>Conclusiones</i>	<i>17</i>

Introducción

El proyecto Agenda consiste en la construcción de un software de escritorio, el mismo propone un programa que ayude al usuario a realizar la administración de sus contactos . Para cumplimentar este fin el cliente final podrá almacenar, editar y borrar los contactos de la agenda.

Para ello se solicita que los mismos posean los siguientes atributos:

- nombre y apellido
- teléfono
- email
- fecha de cumpleaños
- domicilio (calle, altura, piso, dpto y localidad)
- tipo de contacto (familia, amigos , trabajo, etc..)

La agenda posee por defecto una cantidad de ciudades y tipos de contactos determinados. Pero en términos de personalización, el usuario podrá administrar la base de datos agregando, editando y borrando atributos que enriquezcan la agenda.

Además se solicita que el programa refleje unos campos extra para ser asignados a los contactos registrados al momento de crearlos , los mismo albergarán distintas clases de deportes, como así también equipos de fútbol.

Finalmente la aplicación tendrá disponible la posibilidad de realizar un reporte. El mismo consiste en agrupar los contactos registrados por equipo de fútbol, ordenados alfabéticamente, de este modo se podrá llevar adelante una estadística de los mencionados por medio de una planilla y un gráfico.

Objetivo

Los objetivos principales del siguiente TP son trabajar en la construcción de un software para poder desarrollar y plasmar diferentes skills en el uso de determinados recursos. Podemos destacar la utilización de una arquitectura MVC, el cual consta de 3 capas o layers, con el fin de poder separar así las distintas partes del programa. De este modo, describimos:

- Capa vista que representa la interfaz donde el usuario va a poder interactuar con la aplicación
- Capa controlador que va a cumplir la función de intermediario, comunicando las peticiones del front y ejecutando el código correspondiente del back
- Capa modelo, donde principalmente vamos a tener la estructura de los objetos que posee el programa y así también la ejecución de las queries necesarias para interactuar con la base de datos.

Además debemos destacar la utilización de los siguientes patrones de diseño

- Singleton nos permite asegurarnos que una clase tenga una única instancia, de este modo nos proporciona un punto de acceso global a dicha instancia
- patrón DTO, tiene como fin crear un objeto con una serie de atributos que puedan ser enviados o recuperados desde el servidor por medio de una llamada, de esta forma podemos concentrar en una única clase información de diferentes tablas de la base de datos
- patrón DAO, es el que encapsula toda la lógica de acceso de datos y lo separa del resto de la aplicación. de esta manera dao nos proporciona los métodos necesarios para interactuar directamente con la base de datos

Con respecto a la parte del servidor, se requiere el uso de una base de datos de manera local. Crear las tablas, los datos que la componen y normalizarlas de manera correcta para llegar a consumir las mismas desde la aplicación

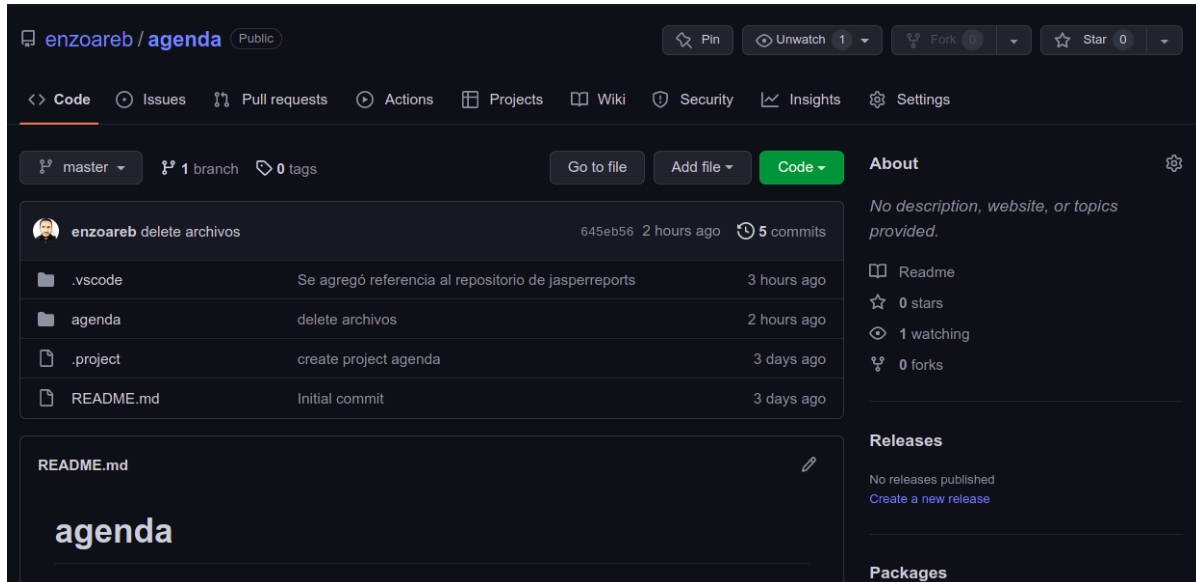
Utilizaremos una librería llamada Jasperreports con la cual podremos crear reportes que recuperen los datos almacenados y mostrarlos en forma de estadística como texto plano o gráficos

Por otro lado señalamos la utilización de un repositorio remoto para el correcto trabajo de los integrantes del equipo donde se pueda almacenar, consumir las diferentes versiones y trabajar con ramas en paralelo.

Finalmente crear un ejecutable en el que almacenamos todas las tecnologías utilizadas y necesarias para el correcto funcionamiento de la aplicación, de esta manera facilitaremos su uso e instalación por parte del usuario final.

Configuración de entorno

En una primera fase se determinan las tecnologías y el ecosistema a utilizar en el proyecto, por lo cual destacamos la utilización de la herramienta de control de versiones que nos brinda GIT, a través de su plataforma GITHUB. Con la misma podremos crear un repositorio online en el cual tener un seguimiento del proyecto y poder trabajar sobre él con ramas locales.



captura de pantalla del repositorio remoto github

Enlace al proyecto: <https://www.github.com/enzoareb/agenda>

Por otro lado se utiliza la base de datos MYSQL, como factor de persistencia allí se almacenará los datos de los contactos que se ingresen a la agenda

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Re
<input type="checkbox"/> deportes	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> domicilio	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	11	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> equipos	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> localidad	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	6	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> pals	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	4	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> personas	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	10	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> provincia	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	23	InnoDB	utf8mb4_general_ci	16.0 KB	
<input type="checkbox"/> tipocontacto	★ Examinar Estructura Buscar Insertar Vaciar Eliminar	3	InnoDB	utf8mb4_general_ci	16.0 KB	
8 tablas	Número de filas	64	InnoDB	utf8_bin	128.0 KB	

Captura de pantalla de la base de datos agenda y sus tablas

A continuación se baja el proyecto base al repositorio local y se encuentran errores de configuración para poder ejecutar el mismo

Entre las configuraciones que debemos editar mencionamos que se debió agregar al archivo pom.xml:

- versión del SDK
- versión de MYSQL
- la dependencia del archivo org.slf4j

```

<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.7</version>
</dependency>

```

Así también se completo:

- El nombre de los alumnos del grupo y correo electrónico.
- referencia al repositorio de jasperreports. Este se utilizará para realizar los reportes requeridos.

```

92
93   <repositories>
94     <repository>
95       <id>jasperreports</id>
96       <url>http://jasperreports.sourceforge.net/maven2</url>
97     </repository>
98     <repository>
99       <id>jaspersoft-third-party</id>
100      <url>https://jaspersoft.jfrog.io/jaspersoft/third-party-ce-artifacts/</url>
101    </repository>
102  </repositories>
103

```

Se creó la base de datos por medio del siguiente script

```

CREATE DATABASE `agenda`;
USE agenda;
CREATE TABLE `personas`
(
  `idPersona` int(11) NOT NULL AUTO_INCREMENT,
  `Nombre` varchar(45) NOT NULL,
  `Telefono` varchar(20) NOT NULL,
  PRIMARY KEY (`idPersona`)
);

```

agregando así la primera tabla Persona, con los campos mínimos establecidos. una vez realizado se procede a agregar la referencia a la base de datos para poder establecer la conexión con la misma

Con estas configuraciones el proyecto se pudo levantar en los repositorios locales para poder trabajar sobre ellos

Base de datos

A Continuación se describe las estructura utilizadas para crear las tablas en la base de datos que albergarán los datos solicitados por el cliente

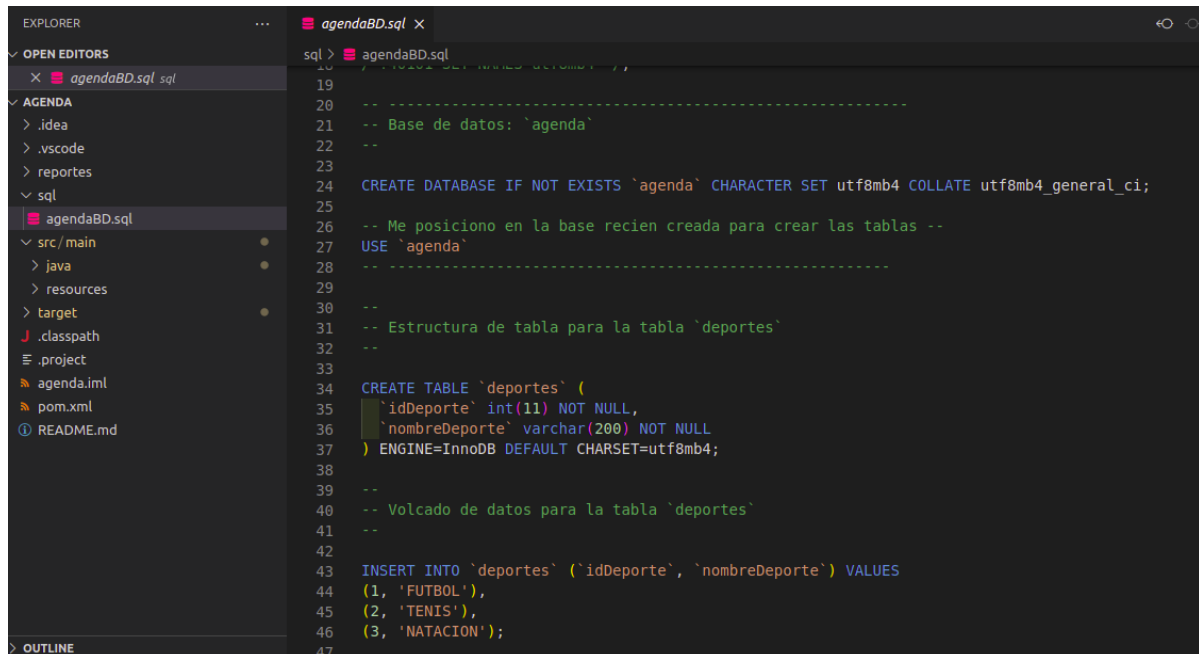
- DOMICILIO
 - Id domicilio (PK)
 - Id persona (FK)
 - calle
 - Altura
 - piso
 - dpto
 - Id localidad (FK)
- LOCALIDAD
 - Id Localidad (PK)
 - nombre
 - Id provincia (FK)
 - Id pais (FK)
- PAÍS
 - Id Pais (PK)
 - Nombre
- PERSONAS
 - Id persona (PK)
 - nombre
 - teléfono
 - email
 - fecha cumpleaños
 - Id tipo de contacto (FK)
 - Id deporte (FK)
 - Id equipo (FK)
- PROVINCIA
 - Id provincia (PK)
 - Nombre
- TIPO DE CONTACTO
 - Id tipo de contacto (PK)
 - nombre

- DEPORTE
 - Id deporte (PK)
 - nombre
- EQUIPO
 - Id equipo (PK)
 - Nombre

Datos importantes de la base de datos:

Como se informó anteriormente se completa la estructura de la base de datos agregando las PK Y FK necesarias para mantener la relación adecuada entre las tablas.

Se crea un script y se lo sube al proyecto, como guía de la estructura necesaria a crear



The screenshot shows a code editor with a file explorer on the left and a SQL script in the main editor. The file explorer shows a project named 'AGENDA' with various files and folders. The SQL script is named 'agendaBD.sql' and contains the following code:

```

19  -- Base de datos: 'agenda'
20  --
21  -- Base de datos: 'agenda'
22  --
23  --
24  CREATE DATABASE IF NOT EXISTS `agenda` CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
25  --
26  -- Me posiciono en la base recién creada para crear las tablas --
27  USE `agenda`
28  --
29  --
30  --
31  -- Estructura de tabla para la tabla `deportes`
32  --
33  --
34  CREATE TABLE `deportes` (
35    `idDeporte` int(11) NOT NULL,
36    `nombreDeporte` varchar(200) NOT NULL
37  ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
38  --
39  --
40  -- Volcado de datos para la tabla `deportes`
41  --
42  --
43  INSERT INTO `deportes` (`idDeporte`, `nombreDeporte`) VALUES
44  (1, 'FUTBOL'),
45  (2, 'TENIS'),
46  (3, 'NATACION');
47

```

captura de pantalla del script almacenado dentro del proyecto

Desarrollo

En este apartado se explicará el código para llevar a cabo el programa, para ello se decide hacer una presentación de los paquetes y las clases que los componen con una breve descripción del código implementado en cada una de ellas

- **paquete DTO:**

contiene los modelos de los objetos según la estructura de datos que tendrán en la base de datos

- class PersonaDTO: se completa los atributos de la clase persona y se agregan los getters y setters
- class DomicilioDTO: se completa los atributos de la clase domicilio y se agregan los getters y setters
- Class DeporteDTO: se completa los atributos de la clase deporte y se agregan los getters y setters
- Class equipoDTO: se completa los atributos de la clase equipo y se agregan los getters y setters
- Class LocalidadDTO: se completa los atributos de la clase localidad y se agregan los getters y setters
- Class provinciaDTO: se completa los atributos de la clase provincia y se agregan los getters y setters
- Class tipoContactoDTO: se completa los atributos de la clase tipo de contacto y se agregan los getters y setters
- Class LocalidadProvinciaDTO: esta clase contiene la estructura de las tablas utilizadas en el programa, conteniendo así los datos necesarios dentro de ella
- Class PersonaDomicilioDTO: esta clase contiene la estructura de las tablas utilizadas en el programa, conteniendo así los datos necesarios dentro de ella

- **paquete main**

- Class main: clase principal para ejecutar el programa crea el objeto controlador

- **paquete modelo**

este paquete conecta las llamadas del controlador con la capa de persistencia

- class agenda: se agregan las funciones necesarias para crear,

editar, listar, buscar y borrar cada uno de los objetos DTO. dsa

- **paquete persistencia**

- **Paquete Conexión**

- Class conexion: aqui se establece la ruta y las credenciales necesarias para generar la conexión con la base de datos

- **Paquete DAO**

- **Paquete interfaz**

este paquete tiene las interfaces de las funciones a implementar por el capa de persistencia de datos

- interface personaDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - interface domicilioDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - interface deporteDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - Interface equipoDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - Interface localidadDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - Interface provinciaDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - Interface paisDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - Interface tipocontactoDAO: se crean las funciones insert-edit-delete-readAll que toman como parámetro un objeto DTO
 - interface LocalidadProvinciaDAO se crea las funcion readAll que devuelven como parámetro una lista de objetos DTO
 - Interface PersonaDomicilioDAO se crea las funcion readAll que devuelven como parámetro una lista de

objetos DTO

■ Paquete mysql

este paquete contiene las querys necesarias para crear los ABM de los objetos de la agenda

- class PersonaDAOsql: implementa las funciones insert, edit, delete y readAll
- class DomicilioDAOsql: implementa las funciones insert, edit, delete y readAll
- class LocalidadDAOsql: implementa las funciones insert, edit, delete y readAll
- class ProvinciaDAOsql: implementa las funciones insert, edit, delete y readAll
- class PaisDAOsql: implementa las funciones insert, edit, delete y readAll
- class tipoContactoDAOsql: implementa las funciones insert, edit, delete y readAll
- class DeporteDAOsql: implementa las funciones insert, edit, delete y readAll
- class EquipoDAOsql: implementa las funciones insert, edit, delete y readAll
- class LocalidadProvinciaDAOsql: implementa las funciones insert, edit, delete y readAll
- class PersonaDomicilioDAOsql: implementa las funciones insert, edit, delete y readAll

● paquete presentacion

○ Paquete reporte

- Class reporteDeporte carga el reporte y lo muestra por pantalla

○ Paquete controlador

este paquete se encarga de conectar la interfaz con el código de negocio y persistencia de la aplicación

- class controlador: carga los componentes del programa, escucha los llamados de los eventos ejecutados en las distintas pantallas del sistema. A su vez realiza los llamados a las funciones correspondientes del código de negocio

● paquete presentación

○ Paquete vista

- class vista: se agregan los atributos de los objetos persona

necesarios a mostrar y los botones necesarios para realizar el ABM de los contactos, localidades, tipo de contacto y mostrar el reporte

- class ventanaPersona: en esta ventana se visualizan los campos para crear un nuevo contacto, también funciona para editar un contacto ya existente, trayendo todos sus datos y mostrarlos para completar su actualización
- class ventanaLocalidad: se crea esta ventana para mostrar las localidades que se encuentran en la base de datos, también el botones correspondiente para realizar el ABM
- class ventanaEditarLocalidad: se crea esta ventana para crear una nueva localidad o actualizar una ya existente
- class tipoDeContacto se crea esta ventana para mostrar los tipos de contacto que se encuentran en la base de datos, también el botones correspondiente para realizar el ABM
- class editarTipoDeContacto: se crea esta ventana para crear un nuevo tipo de contacto o actualizar una ya existente
- class ventanaConexion esta vista aparece al iniciar el programa, es para establecer los el usuario y la contraseña para generar la conexión con la base de datos

Interfaz de usuario

Pantalla Conexión:

CONEXION DEL SERVIDOR ✕

Servidor

localhost

Usuario

root

Contraseña

Conectar

Pantalla Principal

AGENDA													
Id	Nombre y ap...	Telefono	Email	Cumpleaños	Calle	Altura	Piso	Depto	Localidad	Contacto	Deporte	Equipo	
65	jose fonseca	112345678	jose@gmail.c...	12/01/1987	san martin	123	1	b	MALVINAS	FAMILIA	FUTBOL	RIVER PLATE	
11	Martin Rodrig...	1563232565	martin@gmai...	21/09/1971	JOSE C PAZ	3841	1	2	SAN MIGUEL	AMIGOS	NATACION	RIVER PLATE	
9	Walter Perez	1565252565	walter@gmai...	12/12/1971	SAN LUIS	1095	1	8	SAN MIGUEL	AMIGOS	NATACION	RIVER PLATE	
5	Ivan Quiñones	1159999999	ivan@gmail.c...	30/08/2010	SAN LUIS	1541	1	5	SAN MIGUEL	TRABAJO	NATACION	RIVER PLATE	
2	jose Luis Qui...	155562999	luis@gmail.c...	01/12/1971	SAN JOSE	1095	1	8	SAN MIGUEL	TRABAJO	FUTBOL	BOCA JUNIORS	
75	Ivanna ares	12345678	gbiddiddi@g...	28/01/1986	mazzini	2150	7	3	SAN MIGUEL	FAMILIA	NATACION	BOCA JUNIORS	
4	Sabrina Jofre	1568049999	sabrina@gm...	12/12/1980	SAN LUIS	1095	1	8	SAN MIGUEL	TRABAJO	FUTBOL	SAN LORENZO	
3	Enzo Arebalos	1565323259	enzo@gmail...	18/8/1990	Saporiti	2315	1	6	HURLINGHAM	TRABAJO	FUTBOL	SAN LORENZO	
3	Enzo Arebalos	1565323259	enzo@gmail...	18/8/1990	Peron	2315	2	1	SAN MIGUEL	TRABAJO	FUTBOL	SAN LORENZO	
67	gerardo	12345678	gerardo@gm...	11/12/1993	mustoni	453			HURLINGHAM	FAMILIA	TENNIS	SAN LORENZO	
66	roberto arbalo	112345678	roberto@gm...	12/09/1992	robles	231	PB		SAN MIGUEL	AMIGOS	TENNIS	INDEPENDIE...	

Agregar Contacto

Editar Contacto

Borrar Contacto

Localidades

Tipos Contacto

Emitir Reporte

pantalla agregar nuevo contacto

NUEVO CONTACTO ✕

Nombre y ape...

Email

Telefono

Cumpleaños

Domicilio

Calle

Piso

Altura

Dpto

Localidad

SAN MIGUEL

Tipo Contacto

TRABAJO

Deporte

FUTBOL

Equipo Futbol

RIVER PLATE

Guardar

pantalla de localidades

LOCALIDADES			
Id	Nombre	Provincia	Pais
1	SAN MIGUEL	BUENOS AIRES	ARGENTINA
2	MALVINAS	BUENOS AIRES	ARGENTINA
3	HURLINGHAM	BUENOS AIRES	ARGENTINA
4	Lanus	BUENOS AIRES	ARGENTINA
5	carlos paz	CORDOBA	ARGENTINA
6	bariloche	RIO NEGRO	ARGENTINA

Agregar

Editar

Borrar

pantalla para editar localidades

NUEVA LOCALIDAD

Nombre

Provincia

BUENOS AIRES

Pais

ARGENTINA

Guardar

pantalla tipo de contacto

TIPOS DE CONTACTO	
Id	NombreTipo
1	TRABAJO
2	FAMILIA
3	AMIGOS

Agregar

Editar

Borrar

pantalla para editar tipo de contactos

NUEVO TIPO CONTACTO

Nombre

Guardar

Reporte

El reporte tiene como enunciado poder agrupar los contactos almacenados por equipo de fútbol y así también ordenarlos alfabéticamente. Para ello se procedió a generar el mismo con la herramienta Jaspersoft Studio, creando un archivo en blanco, luego vinculándolo con la base de datos y generando las consultas para que traiga los datos a plasmar. Una vez terminada su confección se compila y se importan los archivos ReporteDeporte.jasper y ReporteDeporte.jrxml al proyecto. Para su implementación se creó la clase ReporteDeporte el cual es utilizado como origen de datos para su visualización en el reporte.

El reporte es invocado desde la vista a través de la clase controlador invocando la función mostrarReporteDeporte().

Page 1 of 3 lunes 05 septiembre 2022

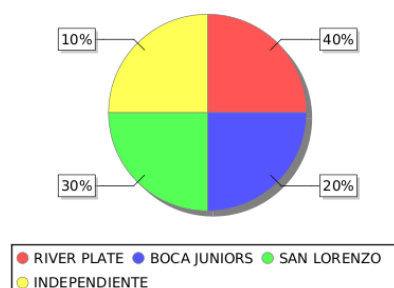
Listado de contactos
Agrupados por Equipo
Ordenados Alfabeticamente

Id Contacto	Nombre	Telefono	email	Deporte preferido
BOCA JUNIORS				
2	Jose Luis Quiñones	155562999	luis@gmail.com	FUTBOL
Total				1
INDEPENDIENTE				
66	roberto arbalo	112345678	roberto@gmail.com	TENNIS
Total				1
RIVER PLATE				
5	Ivan Quiñones	1159999999	ivan@gmail.com	NATACION
Total				1

Captura de pantalla del reporte hoja1

A continuación se muestra la captura de pantalla de la hoja del reporte con un gráfico que muestra cantidad de personas por equipos de fútbol.

Porcentaje de contactos por Equipo de Fútbol



Instalador

Para construir el instalador se utilizó el programa NSIS, con el mismo se pudo generar el ejecutable configurando los programas necesarios para que el software trabaje adecuadamente.

Para generar el ejecutable utilizamos como punto de partida el script proporcionado por los docentes. Se agregó en el script la siguiente sección:

```
#Seccion Prerequisitos, ejecución de otros instaladores
```

```
Section "Prerequisitos" prerequisites
```

```
SectionIn RO
```

```
; requisitos para instalar Mysql
```

```
DetailPrint "Comenzando la instalación de .Net Framework 4.0"
```

```
File "dotNetFx40_Full_x86_x64.exe"
```

```
ExecWait "$INSTDIR\dotNetFx40_Full_x86_x64.exe"
```

```
DetailPrint "Comenzando la instalación del Paquete redistribuible de Visual C++ 2013"
```

```
File "vcredist_x64.exe"
```

```
ExecWait "$INSTDIR\vcredist_x64.exe"
```

```
DetailPrint "Comenzando la instalacion de Mysql Server"
```

```
File "mysql-installer-community-5.6.26.0.msi"
```

```
ExecWait "'msiexec' /i '$INSTDIR\mysql-installer-community-5.6.26.0.msi' /passive'
```

```
DetailPrint "Comenzando la instalacion de Java"
```

```
File "jdk-18_windows-x64_bin.exe"
```

```
ExecWait "$INSTDIR\jdk-18_windows-x64_bin.exe"
```

```
SectionEnd
```

para instalar todo lo necesario para la ejecución de la aplicación Agenda.

Las herramientas que agregamos son:

- java jdk 18
- .Net framework 4.0
- c++
- mysql
- agenda.jar

una vez terminado se probó en una computadora con una arquitectura x86, el cual funcionó satisfactoriamente.

Dificultades encontradas

A lo largo del proyecto las principales dificultades que tuvimos fueron a nivel de configuración, generando así que procesos simples retrasaron el avance el proyecto, ya sea a nivel de compatibilidad, versiones u otras aristas de las distintas herramientas que utilizamos. Uno de los ejemplos que podemos mencionar es a raíz que la aplicación lo mostraba el reporte, siendo esto generado por no haber puesto correctamente la versión del Jasper en archivo pom.xml

Por otra parte, en cuanto a código no encontramos dificultades, esta partición del programa se llevó adelante de manera eficiente .

Conclusiones

Finalizando este proyecto podemos concluir que hemos aprendido y llevado a la práctica la utilización de patrones de diseño, arquitecturas, librerías y procesos enriquecedores que no solo hacen a escribir código. Diseñar una base de datos, configurar entornos, crear un ejecutable son parte del proceso de la construcción del software y tener estos conceptos sólidos nos permite desarrollar un mejor producto al usuario