|  |
| --- |
| **SPRINT 2: Repositorio de Código - Diseño Base de Datos** |

|  |  |
| --- | --- |
| **Identificación Proyecto** | |
| Nombre Proyecto: | Compras |
| Número Equipo: |  |
| **Integrantes del equipo** | |
| Rol  (Líder-Desarrollador – Cliente) | Nombre |
|  | Jaime Ortiz |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |
| --- |
| **Repositorio de Código GitLab o GitHub** |

Como evidencia del repositorio de código, creado con GitLab o GitHub, además de la URL del repositorio, se debe presentar capturas de pantalla donde se visualicen aspectos:

* Creación del proyecto del repositorio.
* Integrantes del equipo invitados.
* Evidencia de la realización de alguna actualización (commit), donde se visualice la actualización y el historial de actualizaciones (Versiones)

<https://github.com/sakurita17/mintic.git>

|  |
| --- |
| **Diseño de la Base de Datos (Proceso de normalización)** |

Como evidencia del Diseño de la Base de Datos, se debe presentar el proceso de normalización efectuado (Formas Normales).

Formato Compras01: Usuarios

* Código de usuario, Nombre, Correo, teléfono, dirección, área y cargo.

Formato Compras02: Requisición

* Código, área, fecha de pedido, cantidad, unidad y producto.

Formato Compras03: Proveedores

* Código, nombre empresa, RUT, correo electrónico, dirección, teléfono

Formato Compras04: Cotización

* Código, empresa, dirección, teléfono, producto, cantidad, valor unitario, valor total.

Formato Compras05: Orden de Compra

* Código, nombre de empresa, dirección, teléfono, producto, cantidad, valor unitario, %IVA, Sub-total y total

Formato Compras06: Entregas

* Código, fecha de la entrega, estado de la entrega, cometarios, cierre de la compra y radicación.

|  |  |
| --- | --- |
| Tabla | Campos |
| Usuarios | cod\_usuario,nombre,mail,telefono,direccion,area,cargo |
| Proveedores | cod\_prov,nombre\_empresa,rut,mail,dirección,ciudad,telefono |
| Requisicion | cod\_req,cod\_usuario,fecha,cantidad,unidad,producto,aprobacion |
| Cotizacion | cod\_cotizacion,cod\_prov,cod\_req,producto,cantidad,valor\_unitario,%IVA,sub-total,total |
| Orden | cod\_orden,cod\_cotizacion,cod\_proveedor |
| Entregas | cod\_entrega,cod\_orden,fecha,cierre,radicacion |

|  |
| --- |
| **Esquema de la Base de Datos (Código SQL)** |

Como evidencia del Esquema de la Base de Datos, se debe presentar el código SQL de creación de la base de datos.

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema compras

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema compras

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `compras` DEFAULT CHARACTER SET utf8 ;

USE `compras` ;

-- -----------------------------------------------------

-- Table `compras`.`dependencia`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`dependencia` (

`cod\_dependencia` INT NOT NULL,

`nombre\_dependencia` VARCHAR(45) NOT NULL,

PRIMARY KEY (`cod\_dependencia`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`Usuarios`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`Usuarios` (

`cod\_usuario` INT NOT NULL,

`nombre\_usuario` VARCHAR(45) NOT NULL,

`mail\_usuario` VARCHAR(45) NOT NULL,

`telefono\_usuario` VARCHAR(45) NOT NULL,

`direccion\_usuario` VARCHAR(45) NOT NULL,

`cargo\_usuario` VARCHAR(45) NOT NULL,

`dependencia\_cod\_dependencia` INT NOT NULL,

`Usuarioscol` VARCHAR(45) NOT NULL,

PRIMARY KEY (`cod\_usuario`, `dependencia\_cod\_dependencia`),

INDEX `fk\_Usuarios\_dependencia1\_idx` (`dependencia\_cod\_dependencia` ASC) VISIBLE,

CONSTRAINT `fk\_Usuarios\_dependencia1`

FOREIGN KEY (`dependencia\_cod\_dependencia`)

REFERENCES `compras`.`dependencia` (`cod\_dependencia`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`Proveedor`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`Proveedor` (

`cod\_prov` INT NOT NULL,

`nombre\_empresa` VARCHAR(45) NOT NULL,

`rut\_empresa` VARCHAR(45) NOT NULL,

`mail\_empresa` VARCHAR(45) NOT NULL,

`direccion\_empresa` VARCHAR(45) NOT NULL,

`ciudad\_empresa` VARCHAR(45) NOT NULL,

`telefono\_empresa` VARCHAR(45) NOT NULL,

`cod\_usuario` INT NOT NULL,

PRIMARY KEY (`cod\_prov`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`estado`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`estado` (

`cod\_estado` INT NOT NULL,

`nombre\_estado` VARCHAR(45) NOT NULL,

PRIMARY KEY (`cod\_estado`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`Requisicion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`Requisicion` (

`cod\_req` INT NOT NULL,

`fecha\_requisicion` DATE NOT NULL,

`cantidad` VARCHAR(45) NOT NULL,

`unidad` VARCHAR(45) NOT NULL,

`producto` VARCHAR(45) NOT NULL,

`aprobacion` VARCHAR(45) NOT NULL,

`Usuarios\_cod\_usuario` INT NOT NULL,

`estado\_cod\_estado` INT NOT NULL,

PRIMARY KEY (`cod\_req`, `Usuarios\_cod\_usuario`, `estado\_cod\_estado`),

INDEX `fk\_Requisicion\_Usuarios\_idx` (`Usuarios\_cod\_usuario` ASC) VISIBLE,

INDEX `fk\_Requisicion\_estado1\_idx` (`estado\_cod\_estado` ASC) VISIBLE,

CONSTRAINT `fk\_Requisicion\_Usuarios`

FOREIGN KEY (`Usuarios\_cod\_usuario`)

REFERENCES `compras`.`Usuarios` (`cod\_usuario`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Requisicion\_estado1`

FOREIGN KEY (`estado\_cod\_estado`)

REFERENCES `compras`.`estado` (`cod\_estado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`Orden`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`Orden` (

`cod\_orden` INT NOT NULL,

`fecha\_orden` DATETIME NOT NULL,

`Usuarios\_cod\_usuario` INT NOT NULL,

`Usuarios\_dependencia\_cod\_dependencia` INT NOT NULL,

PRIMARY KEY (`cod\_orden`, `Usuarios\_cod\_usuario`, `Usuarios\_dependencia\_cod\_dependencia`),

INDEX `fk\_Orden\_Usuarios1\_idx` (`Usuarios\_cod\_usuario` ASC, `Usuarios\_dependencia\_cod\_dependencia` ASC) VISIBLE,

CONSTRAINT `fk\_Orden\_Usuarios1`

FOREIGN KEY (`Usuarios\_cod\_usuario` , `Usuarios\_dependencia\_cod\_dependencia`)

REFERENCES `compras`.`Usuarios` (`cod\_usuario` , `dependencia\_cod\_dependencia`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`Cotizacion`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`Cotizacion` (

`cod\_cotizacion` INT NOT NULL,

`producto` VARCHAR(45) NOT NULL,

`cantida` INT NOT NULL,

`valor\_unitario` DOUBLE NOT NULL,

`%IVA` DOUBLE NOT NULL,

`subtotal` DOUBLE NOT NULL,

`total` DOUBLE NOT NULL,

`Requisicion\_cod\_req` INT NOT NULL,

`Requisicion\_Usuarios\_cod\_usuario` INT NOT NULL,

`Requisicion\_estado\_cod\_estado` INT NOT NULL,

`Proveedor\_cod\_prov` INT NOT NULL,

`Orden\_cod\_orden` INT NOT NULL,

PRIMARY KEY (`cod\_cotizacion`, `Requisicion\_cod\_req`, `Requisicion\_Usuarios\_cod\_usuario`, `Requisicion\_estado\_cod\_estado`, `Proveedor\_cod\_prov`, `Orden\_cod\_orden`),

INDEX `fk\_Cotizacion\_Requisicion1\_idx` (`Requisicion\_cod\_req` ASC, `Requisicion\_Usuarios\_cod\_usuario` ASC, `Requisicion\_estado\_cod\_estado` ASC) VISIBLE,

INDEX `fk\_Cotizacion\_Proveedor1\_idx` (`Proveedor\_cod\_prov` ASC) VISIBLE,

INDEX `fk\_Cotizacion\_Orden1\_idx` (`Orden\_cod\_orden` ASC) VISIBLE,

CONSTRAINT `fk\_Cotizacion\_Requisicion1`

FOREIGN KEY (`Requisicion\_cod\_req` , `Requisicion\_Usuarios\_cod\_usuario` , `Requisicion\_estado\_cod\_estado`)

REFERENCES `compras`.`Requisicion` (`cod\_req` , `Usuarios\_cod\_usuario` , `estado\_cod\_estado`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Cotizacion\_Proveedor1`

FOREIGN KEY (`Proveedor\_cod\_prov`)

REFERENCES `compras`.`Proveedor` (`cod\_prov`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Cotizacion\_Orden1`

FOREIGN KEY (`Orden\_cod\_orden`)

REFERENCES `compras`.`Orden` (`cod\_orden`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `compras`.`Entregas`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `compras`.`Entregas` (

`cod\_entrega` INT NOT NULL,

`fecha\_entrega` DATE NOT NULL,

`cierre\_entrega` VARCHAR(45) NOT NULL,

`radicación\_factura` VARCHAR(45) NOT NULL,

`Orden\_cod\_orden` INT NOT NULL,

`Orden\_Usuarios\_cod\_usuario` INT NOT NULL,

`Orden\_Usuarios\_dependencia\_cod\_dependencia` INT NOT NULL,

PRIMARY KEY (`cod\_entrega`, `Orden\_cod\_orden`, `Orden\_Usuarios\_cod\_usuario`, `Orden\_Usuarios\_dependencia\_cod\_dependencia`),

INDEX `fk\_Entregas\_Orden1\_idx` (`Orden\_cod\_orden` ASC, `Orden\_Usuarios\_cod\_usuario` ASC, `Orden\_Usuarios\_dependencia\_cod\_dependencia` ASC) VISIBLE,

CONSTRAINT `fk\_Entregas\_Orden1`

FOREIGN KEY (`Orden\_cod\_orden` , `Orden\_Usuarios\_cod\_usuario` , `Orden\_Usuarios\_dependencia\_cod\_dependencia`)

REFERENCES `compras`.`Orden` (`cod\_orden` , `Usuarios\_cod\_usuario` , `Usuarios\_dependencia\_cod\_dependencia`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

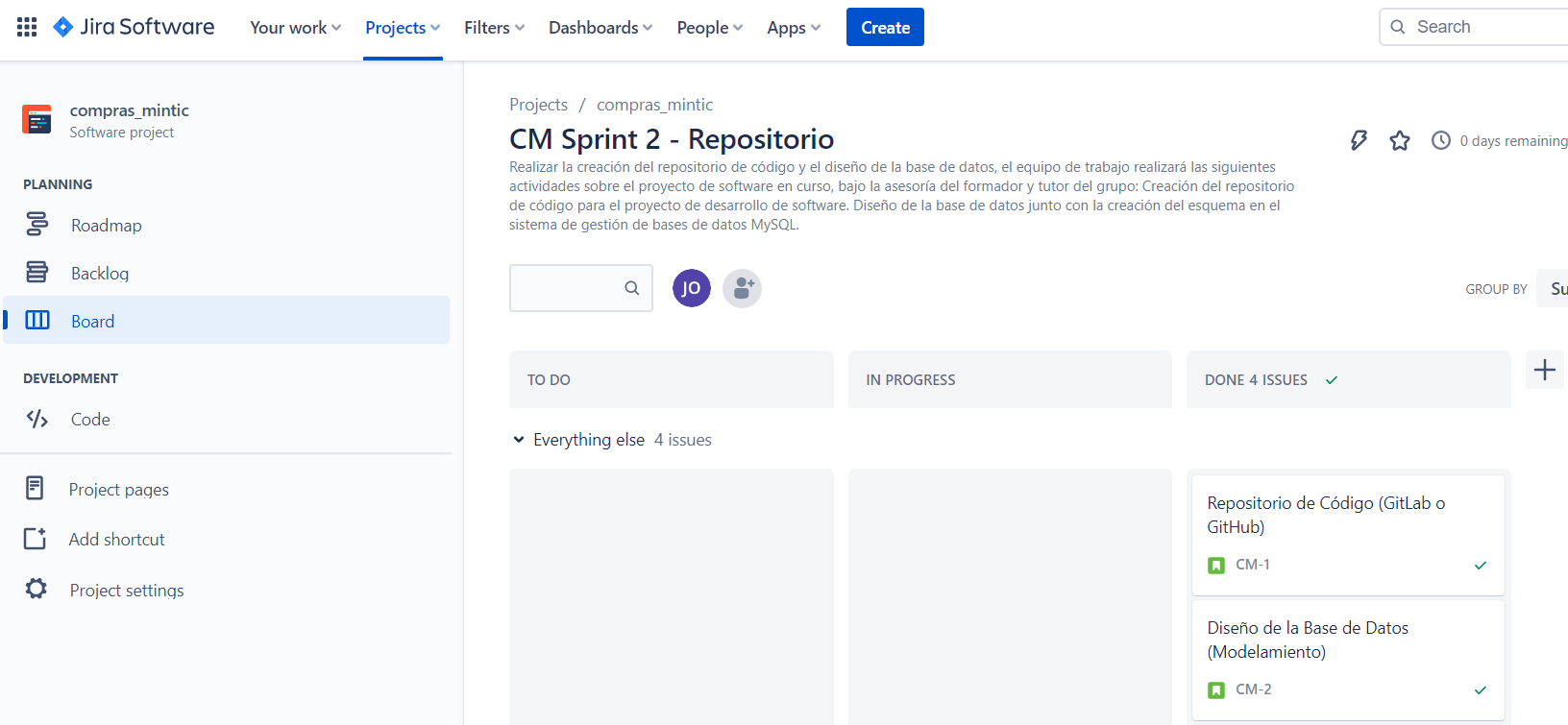
SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

|  |
| --- |
| **Evidencia JIRA (Seguimiento del proyecto)** |

Como evidencia del seguimiento del proyecto con la metodología ágil SCRUM, utilizando el software JIRA, se debe presentar capturas de pantalla donde se visualice la ejecución de los Sprint con las historias de usuario relacionadas con el repositorio de código y el diseño de la base de datos.



|  |
| --- |
| **Evidencias de las Reuniones de Equipo** |

Como evidencia de las reuniones que efectúa el equipo del proyecto, presentar capturas de pantalla de las reuniones efectuadas y si lo consideran pertinente algunas actas de las reuniones.

Estoy trabajando solo