LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

Teoría de Base de Datos

2025

Profesores: Salinas Sergio y Haderne Marisa

Alumna: Nahman Martina

Proyecto Integrador
Plataforma de Gestión de Datos de Sensores para Agricultura de Precisión - "SmartAgro"

Contents

1 Etapa 1: Descripción de la empresa

1.1 SmartAgro

Es un sistema que recopila, almacena, analiza y visualiza datos provenientes de sensores instalados en cultivos y suelos agrícolas. Su objetivo es ayudar a los agricultores a tomar decisiones informadas sobre riego, fertilización, control de plagas y otras prácticas agrícolas, optimizando la producción y reduciendo costos.

La estructura de SmartAgro está organizada en varias áreas clave que permiten cumplir con sus objetivos comerciales y técnicos:

1.1.1 Áreas de la empresa

- **Área Técnica:**
  - **Descripción:** Esta área es responsable de la instalación, mantenimiento y soporte continuo de los sensores y dispositivos tecnológicos utilizados en el campo, tales como sensores de humedad, temperatura, pH y otros dispositivos relacionados con la monitorización de cultivos y suelos. Además, se ocupa de gestionar la infraestructura tecnológica necesaria para procesar, almacenar y garantizar la integridad de los datos recolectados. Este equipo también realiza la actualización de software en los dispositivos y asegura la conectividad de los sensores con la plataforma central de análisis de datos.
  - **Objetivo:** Garantizar que los sensores y equipos estén operativos en todo momento, manteniendo la calidad y precisión de los datos recolectados. Además, asegurar que la infraestructura tecnológica esté optimizada y disponible para el procesamiento de datos sin interrupciones, contribuyendo al funcionamiento eficiente y continuo del sistema SmartAgro.
- **Área de Análisis de Datos:**
  - **Descripción:** Este equipo se encarga de procesar y analizar los datos recolectados por los sensores instalados en los cultivos y suelos. Trabaja en el desarrollo de algoritmos de análisis avanzados, como modelos predictivos y de machine learning, para mejorar la precisión de las predicciones de necesidades de riego, fertilización y control de plagas. También se encarga de generar informes detallados y personalizados para los agricultores, brindando recomendaciones basadas en los datos para optimizar la producción agrícola.

  - **Objetivo:** Transformar los datos crudos en información útil y procesada que permita a los agricultores tomar decisiones informadas. Además, mejorar la precisión de las recomendaciones y predicciones utilizando técnicas avanzadas de análisis de datos, contribuyendo a la optimización de los recursos y a la mejora de los rendimientos agrícolas.

- **Área Comercial y Marketing:**

  - **Descripción:** Esta área tiene la responsabilidad de promocionar y vender los servicios y productos que ofrece SmartAgro. Se encarga de la generación de leads, la gestión de relaciones con los clientes actuales y potenciales, y de desarrollar estrategias de marketing tanto online como offline. También maneja la presencia de la empresa en eventos y ferias del sector agrícola y tecnológico, y realiza campañas publicitarias para aumentar el reconocimiento de marca.
  - **Objetivo:** Ampliar la base de clientes mediante la implementación de estrategias comerciales eficaces y posicionar a SmartAgro como un líder en el sector de la agricultura de precisión. Además, busca aumentar la visibilidad y la percepción positiva de la empresa, creando una imagen sólida en el mercado.

- **Área de Soporte y Capacitación:**

  - **Descripción:** Este equipo proporciona soporte técnico y atención al cliente, respondiendo dudas y resolviendo problemas técnicos relacionados con el uso de la plataforma y los dispositivos. Además, organiza capacitaciones periódicas tanto presenciales como virtuales, con el fin de ayudar a los clientes a entender cómo aprovechar al máximo las funcionalidades de la plataforma y sacar el mayor beneficio de los servicios ofrecidos.
  - **Objetivo:** Garantizar la satisfacción de los clientes mediante un soporte técnico eficiente y personalizado. Además, ofrecer formación continua a los usuarios para que puedan utilizar correctamente los productos y servicios, maximizando así el retorno de inversión y la eficacia de las soluciones implementadas en sus cultivos.

- **Área de Administración:**

  - **Descripción:** Esta área se encarga de gestionar las actividades administrativas de la empresa, incluyendo la contabilidad, la logística y la gestión de recursos humanos. Administra los presupuestos, lleva el control de ingresos y egresos, coordina la adquisición de insumos y equipos, y supervisa los procesos de contratación, capacitación interna y bienestar del personal.

- **Objetivo:** Brindar soporte administrativo integral para asegurar el funcionamiento ordenado y eficiente de la empresa. Facilitar el cumplimiento de los objetivos operativos y estratégicos mediante una gestión adecuada de los recursos financieros, materiales y humanos.

Cada una de estas áreas trabaja de manera interdependiente para alcanzar el objetivo general de SmartAgro: proporcionar una solución integral de monitoreo y gestión para la agricultura de precisión, optimizando los procesos agrícolas y mejorando los rendimientos de los cultivos.

1.1.2 Organigrama

A continuación se muestra el organigrama de la empresa

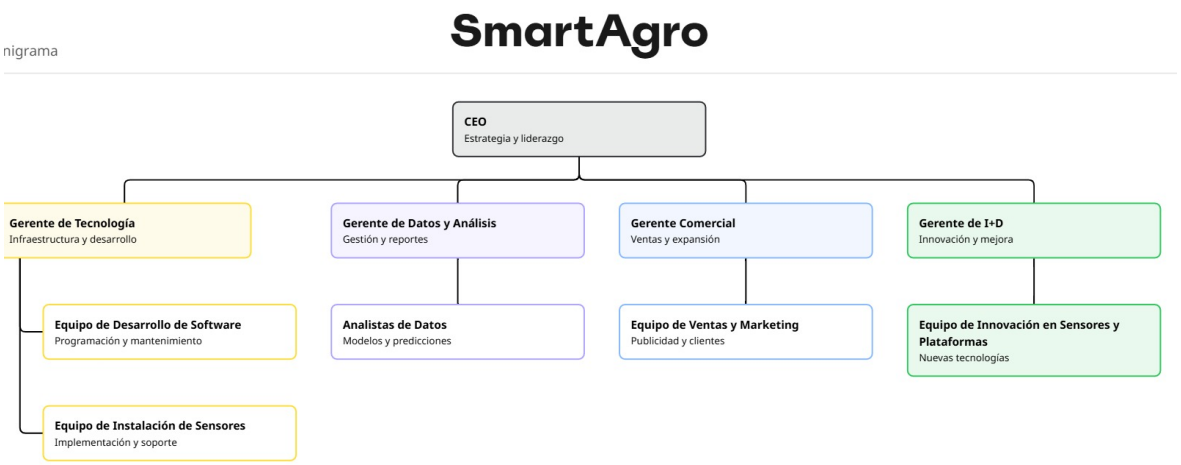

Figure 1: Organigrama SmartAgro

1.1.3 Impacto de SmartAgro

SmartAgro demuestra ser una solución integral para la gestión de datos en la agricultura de precisión, permitiendo a los agricultores tomar decisiones informadas basadas en información en tiempo real. A través de la instalación de sensores avanzados y el desarrollo de una plataforma inteligente de análisis de datos, la empresa ha logrado optimizar el uso de recursos, mejorar la productividad y reducir costos operativos en el sector agrícola.

El procesamiento de variables clave como la humedad del suelo, la temperatura y los niveles de nutrientes ha permitido generar recomendaciones personalizadas, facilitando prácticas más eficientes de riego, fertilización y control de plagas. Además, el asesoramiento técnico y la capacitación brindada a los usuarios han

asegurado una correcta implementación y aprovechamiento de la tecnología en el campo.

En conclusión, SmartAgro no solo representa un avance tecnológico en la gestión agrícola, sino que también contribuye a la sostenibilidad y eficiencia del sector, impulsando una producción más inteligente y adaptada a las necesidades de cada cultivo.

2 Etapa 2: Descripción del sistema a desarrollar

2.1 Sistema a desarrollar

2.1.1 Motivos principales

La gestión eficiente de datos en la agricultura de precisión es fundamental para optimizar recursos, mejorar la productividad y reducir el impacto ambiental. Sin embargo, actualmente muchos agricultores carecen de herramientas tecnológicas accesibles y centralizadas que les permitan monitorear y analizar información clave sobre sus cultivos, como humedad del suelo, temperatura, pH o niveles de riego.

2.1.2 Consecuencias de no implementarlo

La ausencia de un sistema automatizado conlleva una serie de desventajas significativas. Por un lado, el registro manual de datos es propenso a errores humanos y puede llevar a decisiones ineficientes que afectan negativamente la producción. Además, la falta de integración de la información recolectada por los sensores dificulta la toma de decisiones en tiempo real, lo que puede resultar en un uso inadecuado del agua, fertilizantes o recursos humanos. En consecuencia, esto impacta directamente en los costos operativos y en la sostenibilidad de las prácticas agrícolas.

2.1.3 Factibilidad del desarrollo

El desarrollo del sistema es altamente factible debido a la disponibilidad de tecnologías maduras y accesibles como PostgreSQL para la gestión de bases de datos y Java para el desarrollo de aplicaciones. Estas tecnologías permiten construir una solución robusta, escalable y de bajo costo, adaptable a distintas realidades productivas. La propuesta es viable tanto desde el punto de vista técnico como económico, y responde a una necesidad real del sector agrícola que requiere digitalizar sus procesos para competir en un mercado cada vez más exigente y tecnificado.

2.1.4 Motivos específicos

- **Falta de integración de datos de sensores en tiempo real:** Actualmente, muchos productores deben revisar los sensores de forma manual o dependen de registros parciales, lo que impide tener una visión global y actualizada del estado de los cultivos. Esta falta de integración dificulta la detección temprana de problemas como el exceso o déficit de riego, cambios bruscos de temperatura o desequilibrios en el pH del suelo. Como resultado, las decisiones agronómicas se toman con información incompleta o desactualizada, lo que puede derivar en pérdidas económicas y baja eficiencia en el uso de recursos.
- **Registro manual propenso a errores:** Muchas tareas se realizan manualmente, lo cual aumenta el riesgo de errores humanos, pérdida de datos y demoras en el análisis de la información. Esto afecta directamente la eficiencia y la calidad del manejo agrícola.
- **Necesidad de una interfaz intuitiva para la gestión de información:** Los usuarios del sistema, como productores o técnicos agrícolas, pueden no contar con formación técnica en informática. Por eso, se requiere una interfaz gráfica clara, amigable y fácil de usar, que permita acceder, cargar y modificar datos sin complicaciones. Una herramienta intuitiva reduce la curva de aprendizaje, mejora la adopción del sistema y garantiza un uso más eficaz de la plataforma.

El sistema **SmartAgro** busca solucionar estos problemas mediante una **base de datos centralizada** y una **aplicación de gestión de datos**.

2.2 Identificación de los usuarios del sistema y sus características

2.2.1 Agricultores y Productores

- Usuarios principales que utilizarán el sistema para visualizar y gestionar datos de sus cultivos.
- Requieren una interfaz sencilla para consultar información y recibir alertas.

2.2.2 Administradores del Sistema

- Usuarios técnicos encargados de gestionar la base de datos y garantizar el funcionamiento del sistema.
- Necesitan acceso a todas las funciones de administración y configuración.

2.3 Requisitos funcionales y no funcionales del sistema

2.3.1 Requisitos funcionales

- **Gestión de sensores y datos agrícolas:** Registrar información sobre humedad, temperatura, pH, etc.
- **Módulo ABM (Altas, Bajas y Modificaciones):** Permitir CRUD sobre al menos UNA tabla de la base de datos.
- **Autenticación de usuarios:** Permitir acceso seguro a la aplicación.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que no están directamente relaciona dos con las funciones que el sistema debe realizar, sino con cómo debe comportarse. Es decir, definen calidad, rendimiento, seguridad, escalabilidad y restricciones tecnológicas del sistema.

- **Seguridad y control de acceso:** Protección de información sensible.
- **Usabilidad:** La interfaz de la aplicación debe ser intuitiva y fácil de usar, permitiendo que usuarios sin conocimientos técnicos puedan interactuar con el sistema de manera eficiente.

2.4 Alcance del sistema

2.4.1 Funciones incluidas

- **Implementación de una base de datos en PostgreSQL con tablas:** La base de datos es el núcleo del sistema, ya que permitirá almacenar y organizar los datos recopilados de los sensores. Incluirá al menos dos tablas principales, como sensores y mediciones, permitiendo el seguimiento histórico y organizado de las variables del entorno agrícola.
- **Aplicación en Java para gestión de datos mediante ABM:** La aplicación permitirá registrar, modificar y eliminar información de los sensores y sus mediciones de manera sencilla y estructurada, facilitando la administración de los datos directamente desde una interfaz amigable.
- **Sistema de autenticación de usuarios:** Permitirá controlar el acceso al sistema, restringiendo las operaciones sensibles solo a usuarios autorizados, lo que garantiza la integridad y la confidencialidad de la información.

- **Registro de información proveniente de sensores:** El sistema almacenará datos como humedad del suelo, temperatura ambiental y niveles de pH, provenientes de sensores ubicados en los cultivos. Estos datos se registrarán automáticamente en la base de datos.
- **Generación de reportes en formato PDF:** La aplicación permitirá exportar reportes personalizados en PDF basados en los datos recolectados por los sensores, facilitando la toma de decisiones y el análisis histórico de las variables.
- **Simulación de comandos enviados a sensores:** Se incluirá una funcionalidad que simule el envío de comandos básicos a los sensores, como activar el riego, calibrar el sensor de temperatura o ajustar umbrales de alerta. Esta funcionalidad, aunque simulada, permite modelar cómo se integrará la interacción con el hardware en futuras etapas.

2.4.2 Funciones excluidas

- **Predicciones avanzadas con IA:** Se encuentra fuera del alcance del desarrollo actual.

3 Etapa 3: Diagrama Entidad Relación de la Base de Datos

3.1 Diagrama Entidad Relación

A continuación se expone el diagrama entidad relación (DER), el cual describe de manera visual las entidades que componen la base de datos y las relaciones entre ellas. Este diagrama es una representación fundamental para entender cómo se estructuran los datos dentro del sistema.

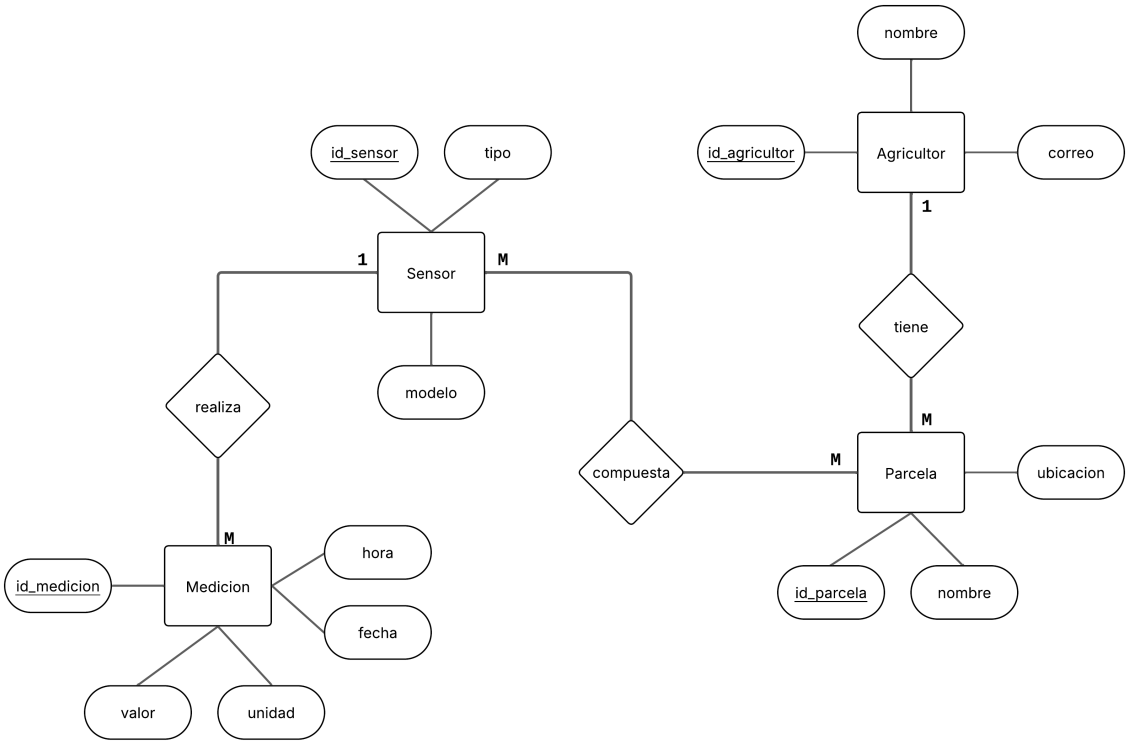

Figure 2: DER

El DER del sistema describe un entorno agrícola donde:
• Agricultores tienen parcelas.
• Cada parcela está compuesta por varios sensores.
• Los sensores realizan mediciones.

3.2 Descripción de las entidades,atributos, claves y relaciones

3.2.1 Entidades

Agricultor

- Atributos: `id_agricultor, nombre, correo`
- Tipo: fuerte
- Clave primaria: `id_agricultor`

Parcela

- Atributos: `id_parcela, nombre, ubicacion`
- Tipo: fuerte
- Clave primaria: `id_parcela`

Sensor

- Atributos: `id_sensor, tipo, modelo`
- Tipo: fuerte
- Clave primaria: `id_sensor`

Medicion

- Atributos: `id_medicion, valor, unidad, fecha, hora`
- Tipo: fuerte
- Clave primaria: `id_medicion`

3.2.2 Relaciones

tiene (entre Agricultor y Parcela)

- Tipo: 1:M
- Un Agricultor puede tener muchas Parcelas
- Cada Parcela pertenece a un Agricultor
- Rol: Un Agricultor tiene Parcelas

compuesta (entre Sensor y Parcela)

- Tipo: M:M
- Una Parcela puede tener muchos Sensores
- Un Sensor puede estar en varias Parcelas
- Rol: Una Parcela tiene Sensores

realiza (entre Sensor y Medicion)

- Tipo: 1:M
- Un Sensor puede realizar muchas Mediciones
- Cada Medicion es realizada por un solo Sensor
- Rol: Un Sensor realiza Mediciones

3.3 Restricciones de integridad referencial

Las restricciones de integridad aseguran la coherencia y validez de los datos en una base de datos. Se refieren a reglas que deben cumplirse para mantener la calidad de la información, evitando datos incorrectos. Incluyen:

- Integridad de entidad: cada entidad debe tener una clave primaria única y no nula.
- Integridad referencial: las claves foráneas deben corresponderse con valores existentes en otras tablas cuando pasemos al modelo relacional

3.3.1 Integridad de entidad

- `id_agricultor, id_parcela, id_sensor e id_medicion` son claves primarias y no pueden ser nulas

3.3.2 Integridad referencial

- Cada Parcela debe tener un Agricultor asociado `(id_agricultor en Parcela referencia a Agricultor)`
- Cada Medición debe estar asociada a un Sensor válido `(id_sensor en Medición referencia a Sensor)`
- En la tabla intermedia `Parcela_Sensor`, las combinaciones `(id_parcela, id_sensor)` deben corresponder a registros válidos en sus tablas respectivas

3.3.3 Otras restricciones

- El correo en Agricultor debe tener un formato válido.
- fecha y hora en Medición deben ser válidas.
- El valor en Medición debe ser un número coherente con la unidad.

3.4 Restricciones de partición

Las restricciones de participación determinan si la participación de una entidad en una relación es obligatoria o no.

- **Participación total:** cada instancia de la entidad debe participar en al menos una instancia de la relación
- **Participación parcial:** algunas instancias de la entidad pueden no participar en ninguna instancia de la relación.

Entidad	Relación	Tipo de participación	Descripción
Agricultor	tiene	Total	Todo agricultor debe tener al menos una parcela.
Parcela	tiene	Parcial	Una parcela puede o no estar asignada a un agricultor en algunos casos
Parcela	compuesta	Total	Cada parcela debe estar compuesta por al menos un sensor.
Sensor	compuesta	Parcial	Un sensor puede no estar asignado a ninguna parcela en algún momento.
Sensor	realiza	Total	Todo sensor debe realizar al menos una medición.
Medición	realiza	Total	Toda medición debe estar asociada a un sensor que la realizó.

Table 1: Relaciones y participaciones de entidades

3.5 Mapeo del DER al modelo Relacional

Agricultor(id_agricultor, nombre, correo)
Parcela(id_parcela, nombreParcela, ubicacion, id_agricultor)
Sensor(id_sensor, tipo, modelo)
Medicion(id_medicion, hora, fecha, valor, unidad,id_sensor)
Compuesta(id_parcela, id_sensor)

Figure 3: Mapeo al modelo Relacional

4 Etapa 4: Diseño conceptual de la Base de Datos

4.1 Dependencias funcionales del modelo

- id_agricultor $\rightarrow$ nombre, correo
- id_parcela $\rightarrow$ nombreParcela, ubicacion, id_agricultor
- id_sensor $\rightarrow$ tipo, modelo
- id_medicion $\rightarrow$ hora, fecha, valor, unidad, id_sensor

- (id_parcela, id_sensor) es clave primaria. No hay más dependencias funcionales.

4.2 Modelo normalizado

Agricultor(id_agricultor, nombre, correo)
Parcela(id_parcela, nombreParcela, ubicacion, id_agricultor)
Sensor(id_sensor, tipo, modelo)
Medicion(id_medicion, hora, fecha, valor, unidad,id_sensor)
Compuesta(id_parcela, id_sensor)

Figure 4: Modelo normalizado

5 Etapa 5: Implementación de la Base de Datos

5.1 Base de datos creada en Postgresql

A continuación se presenta el código implementado en Postgresql

```
-- Crear base de datos
CREATE DATABASE smartagro;
```

5.2 Roles

También se crearon roles para garantizar la seguridad del sistema

```
-- Crear roles
CREATE ROLE admin_smartagro WITH LOGIN PASSWORD 'admin123';
CREATE ROLE agricultor_smartagro WITH LOGIN PASSWORD 'agricultor123';
```

5.3 Esquema

Debido a la poca cantidad de tablas del sistema, opté por sólo crear un esquema llamado "Sistema".

```
-- Crear esquema
CREATE SCHEMA sistema AUTHORIZATION admin_smartagro;
```

5.4 Tablas

Se crearon tablas que garantizan las restricciones referenciales del modelo.

```
CREATE TABLE sistema.Agricultor (
    id_agricultor SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    correo VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE sistema.Parcela (
    id_parcela SERIAL PRIMARY KEY,
    nombreParcela VARCHAR(100) NOT NULL,
    ubicacion TEXT NOT NULL,
    id_agricultor INT NOT NULL,
    FOREIGN KEY (id_agricultor) REFERENCES sistema.Agricultor(id_agricultor)
);

CREATE TABLE sistema.Sensor (
    id_sensor SERIAL PRIMARY KEY,
    tipo VARCHAR(50) NOT NULL,
    modelo VARCHAR(50)
);

CREATE TABLE sistema.Medicion (
    id_medicion SERIAL PRIMARY KEY,
    hora TIME NOT NULL,
    fecha DATE NOT NULL,
    valor NUMERIC(10,2) NOT NULL,
    unidad VARCHAR(20),
    id_sensor INT NOT NULL,
    FOREIGN KEY (id_sensor) REFERENCES sistema.Sensor(id_sensor)
);

CREATE TABLE sistema.Compuesta (
    id_parcela INT NOT NULL,
    id_sensor INT NOT NULL,
    PRIMARY KEY (id_parcela, id_sensor),
```

```
    FOREIGN KEY (id_parcela) REFERENCES sistema.Parcela(id_parcela),
    FOREIGN KEY (id_sensor) REFERENCES sistema.Sensor(id_sensor)
);
```

5.5 Permisos de roles

En base a las tablas creadas anteriormente, se dan permisos a los distintos roles creados.

- **admin_smartagro:** Acceso completo

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA sistema TO
   admin_smartagro;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA sistema TO
   admin_smartagro;
GRANT USAGE ON SCHEMA sistema TO admin_smartagro;
```

- **agricultor_smartagro:** Acceso limitado

```
-- Acceso de lectura a sensores y parcelas
GRANT SELECT ON sistema.sensor TO agricultor_smartagro;
GRANT SELECT ON sistema.parcela TO agricultor_smartagro;

-- Acceso total (insert, update, delete, select) sobre medicion
GRANT SELECT, INSERT, UPDATE, DELETE ON sistema.medicion TO
   agricultor_smartagro;

-- Secuencia de medicion(para insertar nuevos IDs)
GRANT USAGE, SELECT ON sistema.medicion_id_medicion_seq TO
   agricultor_smartagro;

-- Acceso basico al esquema
GRANT USAGE ON SCHEMA sistema TO agricultor_smartagro;
```

5.6 Inserción de datos

A continuación se muestran los datos ingresados en las distintas tablas de la base de datos:

Agricultor:

Query Query History

```
1 SELECT * FROM sistema.agricultor
2 ORDER BY id_agricultor ASC
```

Data Output Messages Notifications

	id_agricultor [PK] integer	nombre character varying (100)	correo character varying (100)
1	1	Juan	juan.perez@email.com
2	2	Maria	maria.gomez@email.com
3	3	Carlos	carlos.diaz@email.com
4	4	Pamela	pamelita.yo@email.com

Figure 5: Datos Tabla Agricultor

Parcela:

Query Query History

```
1 SELECT * FROM sistema.parcela
2 ORDER BY id_parcela ASC
```

Data Output Messages Notifications

	id_parcela [PK] integer	nombreparcela character varying (100)	ubicacion text	id_agricultor integer
1	1	Vid	Norte	1
2	2	Olivo	Norte	2
3	3	Duraznos	Este	3
4	4	Ciruelo	Oeste	4
5	5	Nogales	Sur	4

Figure 6: Datos Tabla Parcela

Sensor:

Query Query History

```
1 SELECT * FROM sistema.sensor
2 ORDER BY id_sensor ASC
```

Data Output Messages Notifications

	id_sensor [PK] integer	tipo character varying (50)	modelo character varying (50)
1	1	Temperatura	T1000
2	2	Humedad	H2000
3	3	Radiacion	R5000
4	4	Temperatura	T2000
5	5	Precipitacion	P3500
6	6	Viento	V4000
7	7	Humedad	H3000

Figure 7: Datos Tabla Sensor

Medicion:

Query Query History

```
1 SELECT * FROM sistema.medicion
2 ORDER BY id_medicion ASC
```

Data Output Messages Notifications

Showing rows: 1 to 7

	id_medicion [PK] integer	hora time without time zone	fecha date	valor numeric (10,2)	unidad character varying (20)	id_sensor integer
1	1	08:00:00	2025-06-10	22.50	°C	1
2	2	08:05:00	2025-06-10	78.30	%	2
3	3	08:10:00	2025-06-10	540.00	W/m²	3
4	4	08:15:00	2025-06-10	25.10	°C	4
5	5	08:20:00	2025-06-10	12.70	mm	5
6	6	08:25:00	2025-06-10	18.40	km/h	6
7	7	08:30:00	2025-06-10	65.90	%	7

Figure 8: Datos Tabla Medicion

Compuesta:

Query Query History

```
1 SELECT * FROM sistema.compuesta
2 ORDER BY id_parcela ASC, id_sensor ASC
```

Data Output Messages Notifications

	id_parcela [PK] integer	id_sensor [PK] integer
1	1	1
2	1	2
3	2	3
4	3	4
5	3	5
6	4	6
7	5	7

Figure 9: Datos Tabla Compuesta

6 Etapa 6: Implementación parcial del sistema

En esta etapa se desarrolló parcialmente el sistema propuesto, abarcando las funcionalidades principales de gestión de datos, exportación y seguridad.

6.1 ABM sobre las tablas Agricultor y Parcela

Se decidió implementar las operaciones de Alta, Baja y Modificación (ABM) sobre las tablas Agricultor y Parcela, ya que ambas entidades son fundamentales para el funcionamiento del sistema. Los agricultores representan a los usuarios principales del sistema, y las parcelas están asociadas directamente a sus actividades.

Para la implementación se utilizó Java junto con Hibernate como framework de persistencia. Hibernate permite mapear las clases Java con las tablas de la base de datos, facilitando así las operaciones CRUD (Create, Read, Update, Delete) de forma sencilla y segura.

Capturas de código utilizando Hibernate:

```java
@Entity
@Table(name = "Agricultor", schema = "sistema")
public class Agricultor {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_agricultor")
    private int id;

    @Column(nullable = false, length = 100)
    private String nombre;

    @Column(nullable = false, unique = true, length = 100)
    private String correo;

    @Column(nullable = false)
    private boolean activo;

    @OneToMany(mappedBy = "agricultor", cascade = CascadeType.ALL, orphanRemoval = true)
    private List<Parcela> parcelas;
```

Figure 10: Agricultor Java Hibernate

```java
@Entity
@Table(name = "Parcela", schema = "sistema")
public class Parcela {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_parcela")
    private int id;

    @Column(name = "nombreParcela", nullable = false, length = 100)
    private String nombreParcela;

    @Column(nullable = false, columnDefinition = "TEXT")
    private String ubicacion;

    @Column(nullable = false)
    private boolean activo;

    @ManyToOne
    @JoinColumn(name = "id_agricultor", nullable = false)
    private Agricultor agricultor;
```

Figure 11: Parcela Java Hibernate

6.2 Implementación del sistema

El desarrollo del sistema se encuentra disponible en el siguiente repositorio de GitHub: https://github.com/martupiru/SmartAgro. Allí se puede acceder al código fuente completo, incluyendo la configuración del entorno, las clases Java utilizadas para la gestión de entidades mediante Hibernate, y otros módulos desarrollados durante esta etapa.