Universidad del Valle de Guatemala
Programación Orientada a los Objetivos
Proyecto Fase 2

Proyecto Finca

Link del github:
https://github.com/MrMenth0l/FincAI/tree/17b8d6090129222dbf4cdfba2de8b15f861df35a/codigo/src

Arana Arrivillaga Andrew Alexander #241018

Hercules Perez Yehosua Friedrich #241452
Rosas Guerra Jose Miguel #241274

Identificación de requisitos funcionales:

El programa debe ser capaz de ayudar de una forma intuitiva a las personas dueñas o que trabajan en las fincas, esto se debe lograr a través del desarrollo de un programa que sea capaz de llevar el control de los diferentes campos que se manejan en las fincas, ya sea el campo agricultor, el ganadero, etc. El programa llevará un control detallado del ganado, con el número de identificación, el peso, el objetivo (carne, leche, etc.), la zona, la salud, la alimentación, etc. Igualmente para el campo agricultor llevaría un control del clima, pudiendo recomendar que tipo de cosecha conviene según el clima, en qué zona de la finca conviene hacerla y todos los factores que cumplirían con el objetivo inicial, facilitar el manejo de las fincas. Paralelamente se llevaría el control de los suministros disponibles siendo capaz de avisar a los trabajadores o al dueño de qué suministros están agotándose, a qué precio están, si su precio tiende a subir o a bajar, comparar precios de diferentes suministradores y cuanto debería comprarse en base a los recursos que se tienen previamente y a la cantidad de ganado/cosecha que haya en ese momento.

Lista de requisitos funcionales:

- Llevar control detallado del ganado, número de identificación, peso, objetivo, zona, salud, alimentación, etc.
- Llevar control de las cosechas, estado, clima, temperatura, etc.
- Recomendar o sugerir que tipo de cosecha conviene según el clima y en qué zona de la finca conviene más hacerla.
- Llevar control de los suministros disponibles
- Avisar a los trabajadores o al dueño cuando un suministro esté por acabarse.
- Comparar precios de los suministros entre los diferentes suministradores.
- Mostrar tendencias de los precios de los suministros.

Priorización de los requisitos funcionales encontrados

Prioridad 1: Más importante, sin ellas el programa no funcionará correctamente ni cumpliría su objetivo básico.

Prioridad 2: Importantes para un mejor programa más no son indispensables.

Prioridad 3: Forman parte de la idea inicial, sin embargo, no estamos seguros de llevarlas a cabo.

- Poder llevar el control del ganado y las cosechas (Cuánto hay, detalles, etc.)
- Poder guardar en memoria todos los datos (exportar a csv)
- Creación de cuentas, usuario, finca, etc.
- Sistema de acceso a la cuenta creada de la finca (contraseña, usuario, etc.)
- Posibilidad de que en la finca puedan operar más usuarios y tengan acceso a toda la información facilitada por el programa.
- Control de suministros.
- Comparación de precios entre suministradores y sugerencias de cuánto y cuándo comprar.

- Presentar tendencia de los precios de los suministros
- Sugerencia según el clima de qué tipo de cosecha conviene hacer.

Identificación y Descripción de Clases necesarias:

Modelo	Vista	Controlador
- CabezaGanado - Cosecha - Finca - Finca - Seccion - Suministro - Usuario - Inventario	- Main	- TokenGen - Export_Csv

.

Explicación del uso de cada clase:

- Main
  - Interactúa con el usuario, mostrará los datos y capturando la entrada
- CabezaGanado
  - Representará la información de las cabezas de ganado
- Cosecha
  - Representa toda información sobre la cosecha de la finca
- Finca
  - Maneja la información general de la finca
- FincaI
  - Almacena la lista de todas las fincas registradas
- Seccion
  - Representa las diferentes secciones de la finca
- Suministro
  - Representa a los proveedores de productos a la finca
- Usuario
  - Representa a los usuarios registrados en la finca
- Inventario
  - Representa todo el inventario que contiene la finca
- TokenGen
  - Es un controlador para la autenticación de usuario
- Export_Csv
  - Es un controlador para la exportación de datos en formato csv

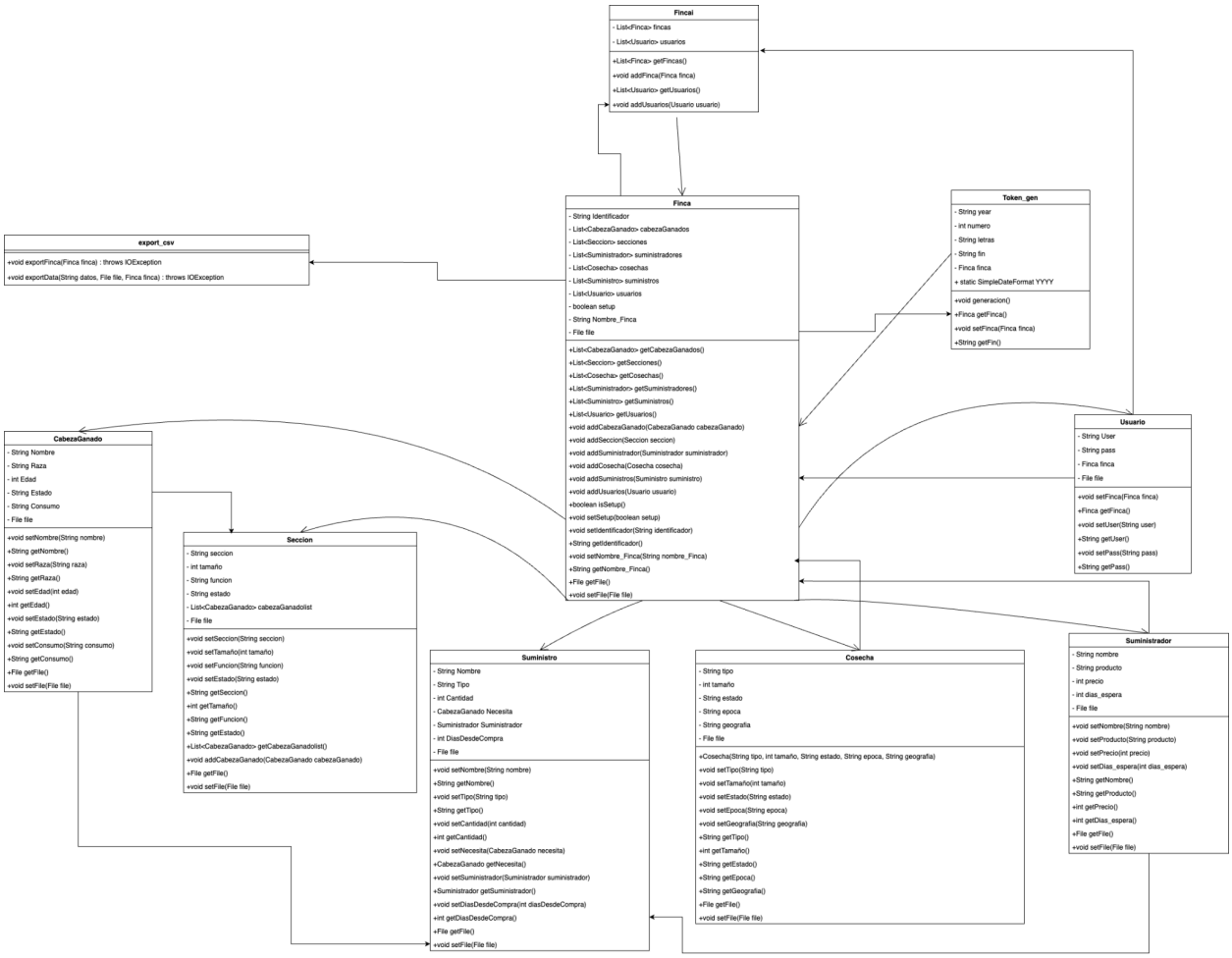
Clase	**Atributo**	**Metodo**
Token_Gen	- year: Año de generación del token. numero: Número de serie o contador para el token.	- generacion(): Genera el token combinando el año, letras del nombre de la finca y un número de serie.

	letras: Letras derivadas del nombre de la finca. fin: Cadena final que representa el token generado. finca: Finca asociada para la generación del token.	getFinca(), setFinca(): Obtiene o establece la finca asociada al token. getFin(): Devuelve el token generado
Clase Usuario	User: Nombre de usuario. pass: Contraseña del usuario. finca: Finca a la que está asociado el usuario.	- generacion(): Genera el token combinando el año, letras del nombre de la finca y un número de serie. getFinca(), setFinca(): Obtiene o establece la finca asociada al token. getFin(): Devuelve el token generado.
Clase Token_gen	Atributos: Nombre Raza Edad Estado Consumo	Métodos: getUser(), setUser(): Obtiene o establece el nombre de usuario. getPass(), setPass(): Obtiene o establece la contraseña del usuario. getFinca(), setFinca(): Obtiene o establece la finca asociada al usuario.
Clase CabezaGanado	Atributos: tipo tamaño estado epoca geografia	Métodos: getNombre(), setNombre() getRaza(), setRaza() getEdad(), setEdad() getEstado(), setEstado() getConsumo(), setConsumo()
Clase Cosecha		Métodos:

		getTipo(), setTipo() getTamaño(), setTamaño() getEstado(), setEstado() getEpoca(), setEpoca() getGeografia(), setGeografia()
Clase Finca	Atributos: Identificador cabezaGanados secciones suministradores cosechas suministros usuarios setup Nombre_Finca	Métodos: addCabezaGanado() addSeccion() addSuministrador() addCosecha() addSuministros() addUsuarios() getCabezaGanados() getSecciones() getSuministradores() getCosechas() getSuministros() getUsuarios() isSetup(), setSetup() setIdentificador(), getIdentificador() setNombre_Finca(), getNombre_Finca()
Clase Main	Atributos: fincas Métodos: getFincas() addFinca()	Métodos: No especificados, pero se encarga de la interacción general con el sistema. Clase Fincai
Clase Seccion	Atributos: seccion tamaño funcion estado cabezaGanadolist	Métodos: setSeccion(), getSeccion() setTamaño(), getTamaño() setFuncion(), getFuncion() setEstado(), getEstado() getCabezaGanadolist() addCabezaGanado()

Clase Suministrador	Atributos: nombre producto precio dias_espera	Métodos: getNombre(), setNombre() getProducto(), setProducto() getPrecio(), setPrecio() getDias_espera(), setDias_espera()
Clase Suministro	Atributos: Nombre Tipo Cantidad Necesita Suministrador DiasDesdeCompra	Métodos: getNombre(), setNombre() getTipo(), setTipo() getCantidad(), setCantidad() getNecesita(), setNecesita() getSuministrador(), setSuministrador()
Clase Fincai	Atributos: fincas: Lista de todas las fincas administradas en el sistema.	Métodos: getFincas(): Devuelve la lista de fincas. addFinca(): Añade una nueva finca a la lista de fincas.
Clase Cosecha	Atributos: tipo: Tipo de cultivo (por ejemplo, maíz, trigo, etc.). tamaño: Área de la cosecha. estado: Estado de la cosecha (por ejemplo, madura, en crecimiento). epoca: Época del año en que se siembra o se cosecha. geografia: Información geográfica del lugar de la cosecha.	Métodos: getTipo(), setTipo(): Obtiene o establece el tipo de cultivo. getTamaño(), setTamaño(): Obtiene o establece el tamaño de la cosecha. getEstado(), setEstado(): Obtiene o establece el estado de la cosecha. getEpoca(), setEpoca(): Obtiene o establece la época de siembra/cosecha. getGeografia(), setGeografia(): Obtiene o establece la información geográfica de la cosecha.

Diseño del Sistema



Investigación de la tecnología disponible

Utilizaremos el lenguaje Java para poder crear una aplicación web que funcione en diferentes sistemas operativos, utilizaremos la herramienta Spring, ya que facilita la gestión de dependencias, nos permitirá poder desarrollar nuestra programación en una página web de un modo fácil y seguro.

Planificación y gestión

Nombre de la tarea	Descripción de la tarea	Horas estimadas de desarrollo	Responsable de desarrollarla	Fecha probable de terminación de la tarea

Crear la clase de Cabeza Ganado	Crear la clase de Ganado para que el usuario pueda ir documentando su cantidad de ganado	8 Horas	Yehosua	13 de agosto
Crear clase Cosecha	Donde el usuario Ingresará la información de la cosecha que tiene	6 Horas	Miguel	14 de agosto
Crear Usuario	Crea la Clase de Usuario para guardar su informacion	6 Horas	Andrew	14 de agosto
Crear clase Finca	Se guarda la informacion acerca de la finca	6 Horas	Yehosua	17 de agosto
Crear clase Suministrador	La clase donde se llevara el nombre de los provedores	5 horas	Miguel	18 de agosto
Crear cuenta Suministro	Guarda información de Inventario	7 horas	Andrew	19 de agosto
Crear clase Seccion	La clase donde divide y explica cada parte de la finca	4 horas	Yehosua	20 de agosto
Crear clase Fincai	Guarda la informacion de toda las fincas registradas	9 Horas	Yehosua y Miguel	27 de agosto
Crear Main	Poder combinar toda las clases y crear la interfaz donde el usuario va a	18 horas	Todos	5 de septiembre

	poder interactuar.			
Crearclase Token_gen	Genera un token para poder ingresar a cada finca (Clave de authenticity)	8 horas	Andrew	7 septiembre
Export_csv	Generar un documento csv con toda la información de la finca	15 horas	Todos	12 de septiembre
Crear Pagina web utilizando spring	Crear Pagina web utilizando spring	72 horas	Todos	30 de sieptiembte

Agosto						
10	11	12	13 Crear la clase de Cabeza Ganado	14 Crear clase Cosecha Crear Usuario	15	16
17 Crear clase Finca	18 Crear clase Suministrador	19 Crear cuenta Suministro	20 Crear clase Seccion	21	22	23
24	24	25	26	27 Crear clase Fincai	28	29
30	31					
Septiembre						

1	2	3	4	5 Crear Main	6	7 Crearclase Token_gen
8	9	10	11	12 Export_csv	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30 Crear Pagina web utilizando spring					

Material a entregar en canvas

- Archivo .pdf que incluya:
o Los requisitos funcionales
o La identificación y descripción de las clases
o La investigación de la tecnología usada
o La planificación de las tareas
o Los formularios de cada uno de los integrantes del grupo
- Archivo de imagen (.jpg o .png) con el diagrama de clases final.
- Archivos de código.
- Link del repositorio github

Evaluación Grupal

- Requisitos funcionales (25 puntos): Se elaboró una lista de los requisitos funcionales a desarrollar en la construcción del sistema. Las funcionalidades descritas resuelven el problema que se detectó. El equipo de desarrollo consideró los requerimientos de seguridad de la herramienta que está desarrollando.
- Identificación y Descripción de clases (10 puntos): El grupo de desarrollo describe correctamente cada una de las clases que identificó, se comprende el propósito de cada clase, de cada atributo y de cada método. La cantidad de clases detectadas es suficiente para resolver el problema planteado. Tiene correspondencia con las clases implementadas

- Diseño del sistema (20 puntos): El sistema se diseñó con el patrón Modelo-Vista-Controlador. Se elaboró el diagrama de clases usando la notación del estándar UML. Las relaciones entre las clases son las adecuadas. Se incluyeron en el diagrama todos los métodos y atributos detectados en la etapa de análisis, incluidos los sets, gets, constructores y toString de cada clase. Están presentes TODAS las clases identificadas.
- Investigación de la tecnología a usar (30 puntos): Se investigó sobre las tecnologías existentes para resolver el problema y selección de la tecnología a usar.
- Planificación (15 puntos): Se elaboró una lista de tareas en la que se reparte de forma equitativa el trabajo entre todos los integrantes del grupo de desarrollo. Todos los integrantes tienen tareas en las que tienen que programar y elaborar parte del análisis y el diseño. Se incluyen las tareas de muestra del sistema a usuarios finales en la planificación de la segunda fase del proyecto.