|  |  |
| --- | --- |
| **Componente de Dominio** | |
| **1. Información General** | |
| **1.1** Identificación | Plantas |
| **1.2** Tipo | Desarrollo propio |
| **1.3** Visión | Es el componente encargado de manejar la información relacionada con las plantas que se siembran en el cultivo, donde se puede identificar algunas características importantes de cada planta. |
| **2. Historia** | |
| **2.1** Versión | 2.0 |
| **2.2** Fecha | 03/11/2019 |
| **2.3** Desarrollador | Sebastian Arboleda/Jonathan Orrego/ Edwin Gomez/ Wilmer Rentería |
| **2.4** Mejoras Principales | Este componente será implementado para manejar todas las características de las plantas ya sea para la APP como para la versión web, componente genérico de creación y consulta de plantas. |
| **3. Metadata** | |
| **3.1** Palabras Claves | Planta, Tipo, Clasificación |
| **4. Funcionalidades básicas / Interfaces** | |
| **4.1 GetPlantList()** | |
| **4.1.1 Nombre** | Obtener las plantas en el sistema |
| **4.1.2 Descripción** | Obtiene un listado de todas las plantas creadas en el sistema |
| **4.1.3 Entradas** | N/A |
| **4.1.4 Salidas** | Listado de plantas |
| **4.2 ManagePlants()** | |
| **4.2.1 Nombre** | Gestión de plantas |
| **4.2.2 Descripción** | Inserta, modifica y borra plantas |
| **4.2.3 Entrada** | Nombre, clasificación y descripción de la planta |
| **4.2.4 Salida** | Booleano indicando el resultado de la operación |
| **5. Calidad** | |
| **5.1 Modificabilidad** | El principal atributo de calidad del componente es su modificabilidad, ya que está creado pensando en el mantenimiento a largo plazo del mismo, manteniendo el CRUD básico de plantas |
| **6. Implementación** | |
| **6.1 Lenguajes** | Java |
| **6.2. Instalación** | Compuesto por varios puntos de fragmentación para las operaciones sobre plantas (B-plant), para instalar se debe implementar una carpeta DAO y realizar el llamado. |

|  |  |
| --- | --- |
| **Componente de Dominio** | |
| **1. Información General** | |
| **1.1** Identificación | VistasBásicas |
| **1.2** Tipo | Desarrollo propio |
| **1.3** Visión | Componente encargado de generar las vistas de la aplicación, menús y en general la navegación de la app. |
| **2. Historia** | |
| **2.1** Versión | 2.0 |
| **2.2** Fecha | 03/11/2019 |
| **2.3** Desarrollador | Sebastian Arboleda/Jonathan Orrego/ Edwin Gomez/ Wilmer Rentería |
| **2.4** Mejoras Principales | Interpolar a más páginas y aplicaciones creando un framework base de desarrollo (headers y contenido) |
| **3. Metadata** | |
| **3.1** Palabras Claves | Inicio, vista, Framework |
| **4. Funcionalidades básicas / Interfaces** | |
| **4.1 GetMapping / RequestMapping** | |
| **4.1.1 Nombre** | Obtener mapeo de rutas |
| **4.1.2 Descripción** | Obtiene el mapeo de rutas a través de un controlador y mapea a una dirección dentro de la aplicación |
| **4.1.3 Entradas** | Dirección página web |
| **4.1.4 Salidas** | Página web |
| **5. Calidad** | |
| **5.1 Modificabilidad** | El principal atributo de calidad del componente es su modificabilidad, ya que está creado pensando en el mantenimiento a largo plazo del mismo, añadiendo controladores y páginas web. |
| **6. Implementación** | |
| **6.1 Lenguajes** | Java, HTML |
| **6.2. Instalación** | Compuesto por varios puntos de fragmentación para el menú (B-menu), y adicional a eso un archivo de customización para colocar el encabezado del nombre de la granja (BCP-farm-name), para instalar se debe implementar una carpeta rest y una carpeta jsp. |

|  |  |
| --- | --- |
| **Pruebas componente de dominio** | |
| **1. Información General** | |
| **1.1** Descripción | Test orientado a probar las salidas de los gráficos para productividad y desperdicio. |
| **1.2** Ambiente de pruebas | Local |
| **2. Casos de prueba** | |
| **2.1** Caso de prueba 001 | |
| **2.1.1** Identificación | testDesperdicios |
| **2.1.2** Descripción | Test para verificar que los desperdicios se mapeen correctamente en el gráfico |
| **2.1.3** Tester | Sebastián Arboleda |
| **2.1.4** Fecha | 2019/11/04 |
| **2.1.5** Archivos relacionados | InventaryResourceTest.java |
| **2.1.6** Entradas | ModelMap |
| **2.1.7** Salidas esperadas | Chart |
| **2.1.8** Salidas reales | Gráfico en forma de listado de los desperdicios |
| **2.1.9** Anomalías | N/A |
| **2.2** Caso de prueba 002 | |
| **2.2.1** Identificación | TestProductividad |
| **2.2.2** Descripción | Test para verificar que la productividad se mapee correctamente en el gráfico |
| **2.1.3** Tester | Sebastián Arboleda |
| **2.1.4** Fecha | 2019/11/04 |
| **2.1.5** Archivos relacionados | InventaryResourceTest.java |
| **2.1.6** Entradas | ModelMap |
| **2.1.7** Salidas esperadas | Chart |
| **2.1.8** Salidas reales | Gráfico en forma de listado de la productividad |
| **2.1.9** Anomalías | N/A |

|  |  |
| --- | --- |
| **Pruebas componente de dominio** | |
| **1. Información General** | |
| **1.1** Descripción | Test orientado a probar las salidas de los reportes de la aplicación |
| **1.2** Ambiente de pruebas | Local |
| **2. Casos de prueba** | |
| **2.1** Caso de prueba 001 | |
| **2.1.1** Identificación | testReporte |
| **2.1.2** Descripción | Test para verificar que el reporte generado cumpla con el mapeo deseado |
| **2.1.3** Tester | Sebastián Arboleda |
| **2.1.4** Fecha | 2019/11/04 |
| **2.1.5** Archivos relacionados | ReporteResourceTest.java |
| **2.1.6** Entradas | ModelMap |
| **2.1.7** Salidas esperadas | String con el reporte |
| **2.1.8** Salidas reales | String con el reporte |
| **2.1.9** Anomalías | N/A |