HISTORIAS DE USUARIO

Presentado por:

Nicolas Quezada Mora - *nquezada@unal.edu.co*

Sharick Yelixa Torres Monroy - *shtorres@unal.edu.co*

Laura Sofia Vargas Rodriguez - *lavargasro@unal.edu.co*

Jeronimo Bermudez Hernandez- jebermudez@unal.edu.co

Profesor:

Oscar Eduardo Alvarez Rodriguez - *oalvarezr@unal.edu.co*

Domingo 09 de Enero

Universidad Nacional de Colombia

Facultad de Ingeniería

Departamento de Industrial y Sistemas

2025

1. Historias de usuario:

Historia de Usuario #6

Descripción conceptual

Módulo	*Módulo de progreso*
Descripción de la(s) funcionalidad(es) requerida(s):	*Se requiere que al final del sprint, se muestre una ventana enseñando el progreso del proyecto*

Backend

Interacción de Componentes:

- **Temporización y Eventos:** Un componente de temporizador detecta la finalización de cada sprint (cada 7 días) para activar el evento de progreso.
- **Persistencia y Gestión de Datos:** Se interactúa con el sistema de guardado/carga para mantener y actualizar el estado del progreso y los objetivos del jugador.
- **Interfaz de Visualización:** La ventana de progreso (UI) recibe y muestra la información actualizada (porcentaje de avance y estado de objetivos), coordinándose con el componente de lógica que procesa estos datos.

Entrada

N/A

Implementación

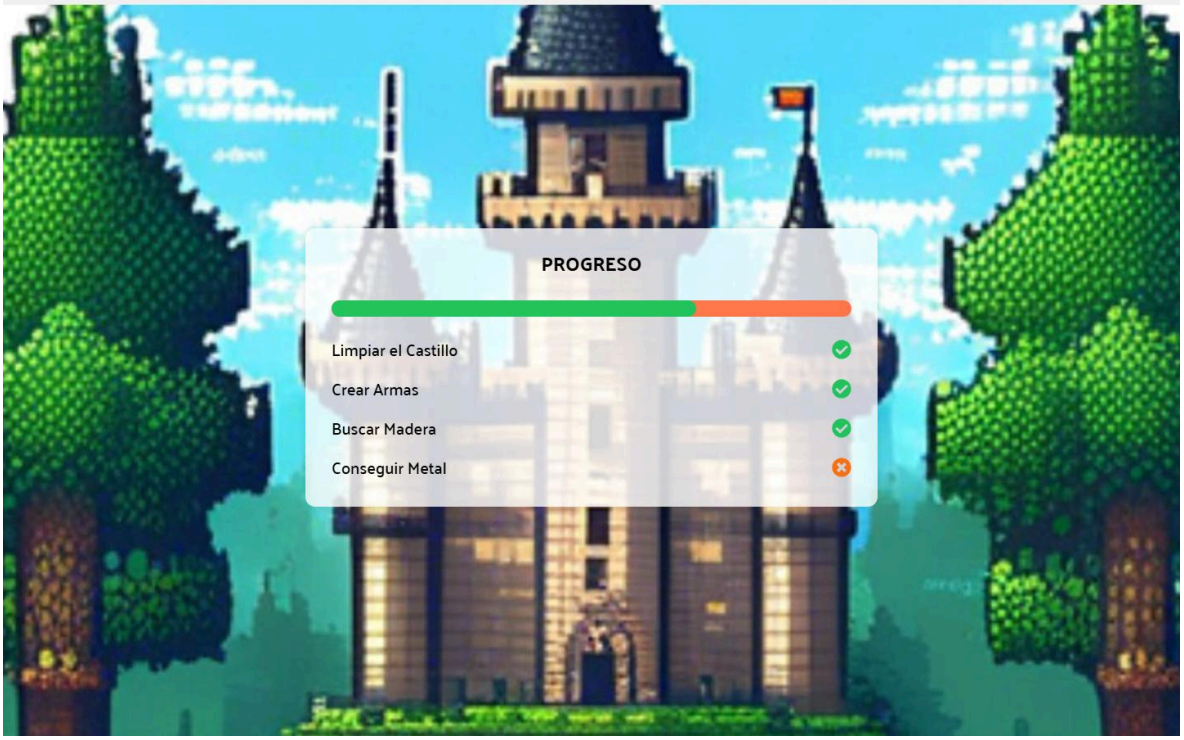
```
public class ProgressManager : Node
{
   private const string SaveFilePath = "user://savegame.json";

   private class SaveData
   {
      public string LastProgressEvent { get; set; }
      public PlayerProgress PlayerProgress { get; set; }
   }

   private class PlayerProgress
   {
      public float TotalProgress { get; set; }
      public List<Objective> Objectives { get; set; }
   }

   private class Objective
   {
      public int Id { get; set; }
```

```
    public string Description { get; set; }
    public string Status { get; set; } // "completed", "inProgress", "failed"
}

private SaveData _currentSaveData;

public override void _Ready()
{
    LoadGameData();
    CheckProgressEvent();
}


private void LoadGameData()
{
    if (File.Exists(SaveFilePath))
    {
        string jsonData = File.ReadAllText(SaveFilePath);
        _currentSaveData =
Newtonsoft.Json.JsonConvert.DeserializeObject<SaveData>(jsonData);
    }
    else
    {
        _currentSaveData = new SaveData
        {
            LastProgressEvent = DateTime.Now.ToString("o"),
            PlayerProgress = new PlayerProgress
            {
                TotalProgress = 0,
                Objectives = new List<Objective>
                {
                    new Objective { Id = 1, Description = "Recolectar 100 monedas", Status =
"inProgress" },
                    new Objective { Id = 2, Description = "Derrotar 10 enemigos", Status =
"inProgress" }
                }
            }
        };
        SaveGameData();
    }
}

private void SaveGameData()
{
    string jsonData = Newtonsoft.Json.JsonConvert.SerializeObject(_currentSaveData);
    File.WriteAllText(SaveFilePath, jsonData);
}

private void CheckProgressEvent()
{
    DateTime lastEvent = DateTime.Parse(_currentSaveData.LastProgressEvent);
    TimeSpan difference = DateTime.Now - lastEvent;

    if (difference.TotalDays >= 7)
```

```
        {
            ShowProgressWindow();
            _currentSaveData.LastProgressEvent = DateTime.Now.ToString("o");
            SaveGameData();
        }
    }

    private void ShowProgressWindow()
    {
        int totalObjectives = _currentSaveData.PlayerProgress.Objectives.Count;
        int completedObjectives = _currentSaveData.PlayerProgress.Objectives.FindAll(obj =>
obj.Status == "completed").Count;
        float progressPercentage = (completedObjectives / (float)totalObjectives) * 100;


        _currentSaveData.PlayerProgress.TotalProgress = progressPercentage;


        var progressWindow = GetNode<ProgressWindow>("ProgressWindow");
        if (progressWindow != null)
        {
            progressWindow.ShowProgress(progressPercentage,
_currentSaveData.PlayerProgress.Objectives);
        }
        else
        {
            GD.Print("Error: No se encontró la ventana de progreso.");
        }
    }

    public void UpdateObjectiveStatus(int objectiveId, string newStatus)
    {
        var objective = _currentSaveData.PlayerProgress.Objectives.Find(obj => obj.Id ==
objectiveId);
        if (objective != null)
        {
            objective.Status = newStatus;
            SaveGameData();
        }
    }
}
```

Frontend

Cada 7 días en el juego, se enseña la ventana de progreso, donde el usuario puede ver su progreso actual, además del estado de los objetivos que tiene pendientes.


PROGRESO
Limpiar el Castillo
Crear Armas
Buscar Madera
Conseguir Metal