UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

Actualización y mejora del software DynVA para medir la agudeza visual dinámica

Trabajo de Fin de Grado
Especialidad de Computación

Entrega 4: Documento final

Diego Delgado Díaz
Director: Luis Belanche
Codirector: Marc Agilés
Tutor GEP: Joan Subirats

Índice de contenido

Índice de figuras

Índice de tablas

1. Introducción y contextualización

La optometría es una disciplina sanitaria, pero no médica, que se encarga del estudio del sistema visual. Las personas que practican un deporte, especialmente a nivel de competición, deben estar en perfectas condiciones físicas para que el cuerpo responda a los estímulos externos de la manera más rápida y precisa posible, y una **buena visión es fundamenta**l para ello. La optometría deportiva trabaja de manera individual con cada persona detectando sus problemas y las necesidades de desarrollo visual que precisa según el deporte que practica y su rol en él.

Hasta hace relativamente poco tiempo no se daba demasiada importancia al cuidado de la visión para mejorar el rendimiento deportivo. Sin embargo, lograr una buena visión puede marcar la diferencia entre ser un buen deportista o ser uno mejor [1].

La agudeza visual es la capacidad del sistema de visión para percibir, detectar o identificar objetos especiales con unas condiciones de iluminación buenas. Para una distancia al objeto constante, si el paciente ve nítidamente una letra pequeña, tiene más agudeza visual que otro que no la ve.

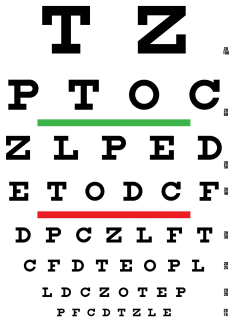


Figura 1: *Test Snellen, utilizado actualmente para determinar la agudeza visual[2].*

La **agudeza visual dinámica** se define como la habilidad de discriminar detalles de un objeto cuando existe movimiento relativo entre dicho objeto y el observador [3].

Este trabajo de fin de grado, pretende desarrollar una aplicación para visualizar una serie de optotipos generados por ordenador con la finalidad de determinar la agudeza visual dinámica de forma objetiva, válida y fiable.

1.1. Contexto

Dentro del Grado en Ingeniería Informática, impartido por la Facultad de Informática de Barcelona, existen diferentes menciones. Este trabajo de fin de grado pertenece a la mención de computación, concretamente, al ámbito de los gráficos por ordenador o computación gráfica.

El proyecto parte del artículo de la Dra. Quevedo[4], donde se diseñó y posteriormente se analizó cualitativamente la validez y la confiabilidad de un nuevo instrumento asistido

por computadora (DynVA 3.0) para la medición de la agudeza visual dinámica (***DVA***[1]). También, de forma paralela, este proyecto parte de la base de una mejora de *software* que se realizó a inicios del 2017. En esta última versión se utilizaba una tecnología más acorde a la época respecto la anterior versión, pero se dejó a medias por causas desconocidas. En este proyecto se pretende dar una imagen nueva al *software*, haciendo que su uso funcional sea el adecuado.

1.2. Conceptos

A continuación, se definen los conceptos/términos clave de este proyecto, así como sus temas relacionados.

1.2.1. Test *DVA*: (DynVA 3.0)

Este test, parte de la siguiente procedimiento: *DVA* se mide de forma binocular[2] instruyendo a participantes a indicar la orientación percibida de optotipo anillo disco Palomar (Figura 2) con el teclado numérico[4].

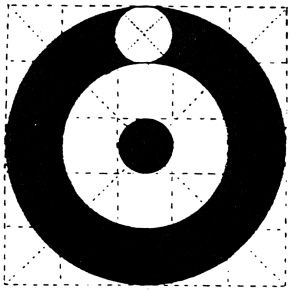


Figura 2: *Optotipo anillo disco Palomar[5], orientación norte.*

El test se puede dividir en dos sub tests básicamente, pero parten de la misma premisa, el optotipo se mueve de un lado a otro, en línea recta o en parábola y la orientación de la figura es al azar entre las 8 disponibles. Los nombres y diferencias de cada subtest son:

- **Speed series**: El optotipo mantiene la velocidad, y a cada intervalo de tiempo t, el **optotipo cambia de tamaño**.
- **Size series**: El optotipo mantiene la velocidad, y a cada intervalo de tiempo t, el **optotipo cambia de velocidad**.

[1] *DVA*: *Dynamic Visual Acuity.*
[2] Tipo de visión en que los dos ojos se utilizan conjuntamente.

1.3. Identificación del problema

En este trabajo se va desarrollar una solución que satisfaga la premisa de poder generar y mejorar el test de Agudeza Visual Dinámica (*DynVA 3.0*) de forma correcta y eficiente. El problema recae básicamente en decidir si se va utilizar la solución del 2017 y corregir los errores de *software* que existen o diseñar la aplicación desde cero.

1.4. Agentes implicados

El sistema a desarrollar está dirigido pero no limitado a deportistas de élite que quieran desarrollar una mejor visión para marcar la diferencia a nivel competitivo. Dicho sistema también puede dirigirse a personas mayores, que por normal general son las más afectadas por trastornos de visón. pero cualquier otra persona fuera de esos subgrupos de la población también se puede beneficiar de esta aplicación.

2. Justificación

La agudeza visual dinámica es una capacidad cuya medición, aun en la actualidad, no suele formar parte de las baterías de test de los centros optométricos, y disponer de aplicaciones de *software* con tecnología actual para poder ayudar a diagnosticar, tratar y mejorar dicha capacidad se ha convertido en una necesidad. En consecuencia, el problema planteado en la sección 1.3, se planteará una serie de argumentos porque es más conveniente desarrollar una nueva aplicación en vez de reutilizar el código existente. También se planteará que lenguaje/tecnología se utilizarán para desarrollar la aplicación.

2.1. Estudio soluciones existentes

La actual aplicación está desarrollada en *Visual Studio*[3] (se desconoce la versión) y esta tecnología está diseñada para aplicaciones de escritorio. Es cierto que se pueden mostrar animaciones generadas por animación, pero el lenguaje no permite abstraerse tanto como otras tecnologías actuales por lo tanto se hace más difícil la lectura y comprensión. De forma alternativa, puede parecer más sugerente y sencillo reutilizar el código fuente, pero no hay ninguna garantía de que sea más fácil corregirlo que empezar de nuevo, así que esta incertidumbre se convierte en el segundo inconveniente.

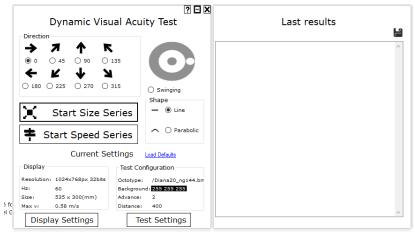


Figura 3: *Panel de control antiguo DynVA 3.0.*

[3] *Microsoft Visual Studio* es un entorno de desarrollo integrado para *Windows* y *macOS*.

3. Alcance

Como en todo proyecto, es importante definir el alcance de este, pues el tiempo de desarrollo es limitado y es necesario definir los objetivos y requerimientos de la aplicación. A continuación, se definen los objetivos, los requerimientos no funcionales y los obstáculos y riesgos del proyecto.

3.1. Objetivos

El objetivo principal de este trabajo es el desarrollo, implementación de los diferentes tipos de test requeridos para evaluar la agudeza visual dinámicos con unos parámetros que variaran como la velocidad/tamaño del test en función de la distancia del usuario a la pantalla o el usuario. Seguidamente, se divide el objetivo en los siguientes subobjetivos:

1. Generar una aplicación que permita generar unos tests visuales desde una interfaz gráfica.
2. Ser capaz de poder producir test visuales a partir de unos parámetros definidos por el usuario.
3. Poder generar una puntuación que mida la agudeza visual dinámica a través de cuántos aciertos ha obtenido el usuario mediante el teclado numérico.

3.2. Requerimientos

3.2.1. Requerimientos funcionales

A pesar de que los principales requerimientos funcionales del sistema se definen en el objetivo, a continuación, se definen otros requisitos necesarios para el funcionamiento de la aplicación;

1. Compatibilidad. Se plantea que la aplicación sea compatible con diferentes tipos de pantalla, tanto en tamaño, resolución o frecuencia de refresco. También se plantea que sea compatible con el mínimo de requerimientos a nivel *hardware* de *Godot* reflejado en la Figura 1.
2. Mantener las funcionalidades presentes en la antigua versión de *software* de *DynVA 3.0*.

3.2.2. Requerimientos no funcionales

A continuación se definen los requisitos no funcionales del sistema, es decir, aquellos requerimientos que no hablan directamente del funcionamiento del sistema, sin embargo, se han de tener en cuenta desde el principio para el desarrollo de la aplicación:

1. Usabilidad. La aplicación ha de ser sencilla de utilizar tanto para el examinador que configura los tests, como para el usuario que quiere ser examinado. Esto requerirá crear una interfaz gráfica de fácil uso.

2. Eficiencia. La aplicación se tiene que poder ejecutar con el mínimo nivel de hardware, por lo tanto se tiene que tener en mente no hacer operaciones computacionales elevadas y no crear código innecesario.

3. Eficacia. La aplicación no debe fallar bajo ningún concepto, debe hacer su tarea lo mejor posible y no dar ningún tipo de error. Esto último puede ser trivial, pero no lo es.

4. Metodología

La metodología ágil de desarrollo de *software* cumple con los requisitos expuestos anteriormente, la idea des definir ciclos de desarrollo cortos donde se diseñe, implemente una funcionalidad.

Concretamente, siguiendo con la metodología ágil y se realizarán reuniones semanales con las personas implicadas en este proyecto, en las que se comprobarán los objetivos propuestos, y en función del estado, se realizarán modificaciones en la planificación y/o se establecerán las nuevas tareas a realizar.

4.1. Herramientas

Para facilitar el seguimiento de los objetivos se usará *Trello*[6], se trata de una herramienta *online* que permite establecer tareas en forma de tarjetas, lista y tableros virtuales. El objetivo es usar un tablero para el proyecto donde una lista representa el grupo de tareas a realizar en una semana, y cada tarjeta es una tarea a realizar.

El control de versiones es una herramienta en cualquier proyecto informático, permite hacer un seguimiento de los cambios del proyecto y sirve como copia de seguridad. Para ello se usará *Github*[7].

Por último, se utilizará la herramienta *Gantter*[8] para la planificación de las tareas. A través del diagrama de *Gantt*, se realizará un control del tiempo dedicado a cada tarea para asegurar la finalización del proyecto en el plazo establecido. En las reuniones semanales, se validará el tiempo gastado en cada tarea y se actualizará el diagrama en consecuencia.

Para el desarrollo de la aplicación es necesario un conjunto de herramientas *software* y *hardware*.

4.1.1. Software

Un *game engine* es un *software* con una interfaz bien definida que permite el desarrollo de videojuegos. Hay varias empresas que se dedican a desarrollar este tipo de soluciones, quizá la más conocida es *Unity*[9] que no es *open source*[4] pero sí es gratuita

[4] *Open source: Software* de código abierto es el *software* cuyo código fuente y otros derechos que normalmente son exclusivos para quienes poseen los derechos de autor, son publicados bajo una licencia de código abierto o forman parte del dominio público.

mientras no se tengan ganancias de más de cien mil dólares por año. *Unity* se desarrolló en el 2005 y se ha utilizada para crear un gran número de videojuegos muy conocidos alrededor del mundo.

Otro *game engine* aparte de *Unity*, sería *Godot*[10] el cuál sí es *open source*. Se estrenó en el 2014 y poco a poco se va haciendo hueco entre sus competidores. Esta tecnología permite hacer *scripts* con un lenguaje propio llamado *GDScript* basado en *Python* y en *Lua*. *Godot* es una herramienta sencilla de utilizar pero también tiene sus inconvenientes: no es ideal para juegos con gráficos complejos en 3D. Este último inconveniente no es un problema ya que las animaciones generadas por el test se desarrollan en un mundo 2D.

Dado que el autor de este TFG es un interesado del mundo *open source* y tiene conocimientos avanzados del lenguaje *Python*, se escoge utilizar *Godot* como herramienta para desarrollar la aplicación.

4.1.2. Hardware

La aplicación será ejecutada en un ordenador, los requerimientos vienen precedidos por la tecnología que escojamos, que en este caso es *Godot*:

CPU	2 *cores*
OS:	*Windows 7* mínimo, *macOS 10.10* mínimo, Linux (64-bit or 32-bit x86)
RAM:	4GB
Tarjeta gráfica	Soporte de *OpenGL 3.3 Core Profile*

Tabla 1: *Requerimientos a nivel hardware de Godot[11].*

Por otro lado, para mostrar los test visuales es necesario una pantalla. También, hará falta un teclado numérico, el cual el usuario utilizará para escoger la dirección de optotipo del test.

5. Planificación temporal

Con el objetivo de finalizar este trabajo de fin de grado en la fecha estimada y cumplir con los objetivos planteados previamente, se realiza una planificación temporal del proyecto dividido en tareas.

El trabajo empieza el día 5 de septiembre y finaliza el 23 de diciembre. En total, el desarrollo del proyecto se llevará a cabo a lo largo de 109 días aproximadamente y unas 327 horas aproximadamente. La dedicación diaria será de 3 horas aproximadamente. Entre semana se quedará con los responsables según disponibilidad horaria y se mostrarán los avances y las mejoras que requiere el proyecto.

En la sección 10 Anexos se muestra el se muestra el diagrama *Gantt* de las tareas (Anexo A).

5.1. Descripción de las tareas

En esta sección se detallan las tareas a realizar de forma individual, pero se agrupan por bloques para distinguir con mayor facilidad las distintas fases del proyecto.

GP - Gestión del proyecto

La gestión del proyecto es esencial para planificar, definir y documentar el trabajo a realizar, además, engloba las reuniones para la validación y propuesta de objetivos semanales. Se estima que que en global el grupo de gestión tendrá una carga de 115 horas.

GP.1 - Alcance

Antes de empezar con el proyecto es necesario acotar el desarrollo, por este motivo, se dedica tiempo inicial a definir qué se quiere conseguir con el trabajo, qué se va a desarrollar y qué medios serán necesarios. La duración ha sido de 5 horas.

GP.2 - Planificación

Para cumplir con los objetivos propuestos en la definición del alcance del proyecto, se realiza una planificación temporal, así como de recursos y requerimientos asociados a cada tarea. Además, se definen los riesgos y obstáculos, y se plantean tareas alternativas para solventarlos. La duración de esta fase ha sido de 10 horas.

GP.3 - Presupuesto

Se realizará un presupuesto para cuantificar el coste del proyecto, para ello, se realizarán partidas por cada tarea teniendo en cuenta los costes de personal y equipo, además, se cuantificarán los costes genéricos y partidas de imprevistos. Se estima una dedicación de 5 horas.

GP.4 - Informe de sostenibilidad

Se analizará a partir de un informe el impacto medioambiental, económico y social del proyecto, en concreto, de las fases de planificación y desarrollo. El tiempo estimado para realizar el informe es de 5 horas.

GP.5 - Reuniones

Como en cualquier proyecto de desarrollo de aplicaciones, es necesario realizar reuniones frecuentemente para analizar los resultados obtenidos según los objetivos semanales, y decidir si se han de cambiar aspectos de la planificación. Se prevén reuniones semanales de 1 hora con los directores del proyecto. En total se estima una duración de 20 horas.

GP.6 - Documentación

Una parte importante del TFG es la memoria final, por lo tanto, a lo largo del desarrollo del proyecto se han de documentar las distintas fases. La documentación se realizará de forma paralela al resto del proyecto, de esta forma se irán incorporando las secciones en las que se trabaje. La duración estimada es de 60 horas.

GP.7 - Presentación

Por último, una vez finalizada la documentación, es necesario preparar la presentación para el tribunal que evaluará el TFG. Se preparará material de soporte para la presentación, así como el guión, y se realizarán ensayos. En total la duración estimada es de 10 horas.

TP - Trabajo previo

En este apartado se especifican las tareas a realizar antes del desarrollo del trabajo, es decir, tareas de preparación y estudio previo. Puesto que se trata de una fase de preparación, se puede realizar paralelamente a la mayoría de las tareas de gestión del proyecto. Se estima una duración de 50 horas.

TP.1 - Aprendizaje

Obtención de los conocimientos necesarios sobre las herramientas y lenguajes involucrados. Aunque ya se posean conocimientos sobre lenguajes de programación, hace falta entender con profundidad:

1. *GDScript*: comprender lenguaje que se utiliza en *Godot*.
2. *Godot*: comprender interfaz que propone dicha herramienta así cómo hacer uso de las técnicas de visionado y animación de objetos.

Con tal de obtener una buena base en estos conceptos se prevé una dedicación de 50 horas al aprendizaje.

DA - Desarrollo aplicación

DA.1 - Diseño

Habiendo decidido las funcionalidades a crear para la aplicación teniendo en cuenta los objetivos establecidos, se para al diseño de estas funcionalidades. Una vez listo, se validará el diseño para que se cumplan los objetivos y garantizar la viabilidad de la solución. Se estiman una duración de 12 horas.

DA.2 - Implementación

La implementación del producto es traducir el diseño de las funcionalidades a código. La programación es la parte más importante y extensa del proyecto. A grandes rasgos se pueden diferenciar la parte gráfica y la mecánica que va por detrás. Es la parte con más peso, con un volumen estimado de 150 horas.

5.2. Recursos

5.2.1. Recursos humanos

En este proyectos se encuentran tres roles diferentes: jefe de proyecto, programador y *tester*. No obstante, teniendo en cuenta que este TFG se realiza por una persona, será el autor el encargado de asumir los diferentes roles en función de la tarea a realizar.

1. Jefe de proyecto: Se encarga de la planificación del proyecto, liderar las reuniones con el equipo y escribir la documentación.
2. Programador: Es la persona encargada de implementar el sistema.
3. *Tester*: Se ocupa de realizar las pruebas de validez del sistema, debe diseñar las pruebas, ejecutarlas y presentar un informe para poder arreglar los errores encontrados.

5.2.2. Recursos materiales

A continuación, se exponen los recursos materiales necesarios para la realización del proyecto.

1. Ordenador para realizar las tareas de gestión y producción del proyecto.
2. *Overleaf* [12]: Entorno de desarrollo para crear documentos con LaTeX[5].
3. *Godot*: Motor gráfico que servirá como entorno de desarrollo para la aplicación.
4. *Gantter*[8]: Herramienta *web* para la creación de diagramas de Gantt.

A continuación se expone la Tabla 2 la asignación de recursos materiales a cada tarea.

Id.	Tareas	Duración	Recursos materiales	Recursos humanos
GP	**Gestión del proyecto**	115h	-	-
GP.1	Alcance	5h	Ordenador	JP
GP.2	Planificación	10h	Ordenador	JP
GP.3	Presupuesto	5h	Ordenador	JP
GP.4	Informe de sostenibilidad	5h	Ordenador	JP
GP.5	Reuniones	20h	Ordenador	JP,T,P
GP.6	Documentación	60h	Ordenador	JP
GP.7	Presentación	10h	Ordenador	JP
TP	**Trabajo previo**	50h	-	-
TP.1	Aprendizaje	50h	Ordenador	P
DA	**Desarrollo aplicación**	162h	-	-
DA.1	Diseño	12h	Ordenador	P
DA.2	Implementación	150h	Ordenador	P,T
-	Total	327h	-	-

JP: Jefe de Proyecto, T: *Tester*, P: Programador.

Tabla 2: *Tabla de tareas con los recursos necesarios.*

6. Gestión del riesgo

Es importante prever los riesgos y obstáculos que pueden surgir durante el desarrollo del TFG. Todo proyecto tienen diferentes dificultades que se deben analizar anticipadamente para minimizar su impacto. Es importante prever los riesgos y obstáculos que pueden surgir durante el desarrollo. A continuación se describen los posibles obstáculos.

1. Dado que el proyecto anterior tiene errores y algunas funcionalidades quedan por añadir, es importante tener una buena comunicación con los solicitantes de este aplicación, no tener una buena comunicación conllevaría un aumento de tiempo al desarrollo.

[5]LaTeX: es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas[13].

2. Cuestiones de código. Un riesgo importante que conlleva el desarrollo de software es la mala calidad de código, se puede mitigar dicho riesgo con la siguientes acciones.
    - Probar el código con frecuencia.
    - Resolver fallos y errores lógicos cuando se encuentran.
    - Utilizar las mejores prácticas de código.
3. Errores en *Godot*: A pesar de que dicha herramienta es utilizada por miles de usuarios, todavía es reciente y pueden contener *bugs*. Si aparece un error que bloquee el desarrollo del proyecto, se tendrá que añadir una tarea a la planificación para desarrollar una solución alternativa a la proporcionada por la librería. Se estima que podría añadir entre 20 y 40 horas de desarrollo.
4. Rendimiento: la aplicación tiene que ser suficiente optimizada para que se pueda ejecutar en equipos con un *hardware* relativamente bajo en especificaciones.

7. Gestión económica

Una vez realizada la planificación temporal del proyecto, se van a estimar los costes necesarios para el desarrollo. Se identifican diferentes tipos de costes asociados al personal, espacio de trabajo, y a las herramientas y dispositivos usados. Además, para superar los obstáculos que aparezcan y asumir los costes no programados, se realiza un plan de contingencia, una partida de imprevistos y se exponen mecanismos para controlar el presupuesto.

7.1. Presupuesto

7.1.1. Costes de personal

A partir de la planificación por tareas se calcula el coste de personal, se tienen en cuenta los 3 roles definidos anteriormente: jefe de proyecto, programador y *tester*. En la Tabla 3 se ve el coste por hora de cada puesto, los datos han sido obtenido de la empresa de reclutamiento *Hays*:

Rol	Coste por hora
Jefe de proyecto	30€/h
Programador	16€/h
Tester	16€/h

Tabla 3: *Costes de personal a partir de la guía de mercado laboral de Hays[14].*

En la Tabla 4 se detallan las partidas por tarea a partir de los costes de personal de la Tabla 3, y se estima el coste de la seguridad social multiplicando el coste por 1,3. En total el coste del personal del proyecto es de 12.847€.

Id.	Tareas	Duración	Roles	Coste	Coste SS
GP	**Gestión del proyecto**	115h	-	4.090€	5.317€
GP.1	Alcance	5h	JP	150€	195€
GP.2	Planificación	10h	JP	300€	390€
GP.3	Presupuesto	5h	JP	150€	195€
GP.4	Informe de sostenibilidad	5h	JP	150€	195€
GP.5	Reuniones	20h	JP,T,P	1.240€	1.612€
GP.6	Documentación	60h	JP	1.800€	2.340€
GP.7	Presentación	10h	JP	300€	390€
TP	**Trabajo previo**	50h	-	800€	1.040€
TP.1	Aprendizaje	50h	P	800€	1.040€
DA	**Desarrollo aplicación**	162h	-	4.992€	6.490€
DA.1	Diseño	12h	P	192€	250€
DA.2	Implementación	150h	P,T	4.800€	6.240€
-	**Total**	**327h**	-	**9.882€**	**12.847€**

JP: Jefe de Proyecto, T: *Tester*, P: Programador.

Tabla 4: *Tabla de partidas por tarea.*

7.1.2. Costes genéricos

A continuación se describen todos los costes independientes a las tareas del proyecto:

- **Espacio físico**: para realizar presencialmente las diversas tareas del proyecto. Se realiza desde el despacho de una vivienda ubicada en Barcelona. El proyecto se desarrolla a lo largo de 4 meses, teniendo en cuenta que cada mes son 90€, el coste final es de 360€.
- **Ordenador**: ya que este proyecto se realiza por una única persona solo es necesario una unidad de ordenador. El ordenador tiene un hardware muy por encima de la media y también sus periféricos.
- **Software**: se utiliza el mismo que se ha mencionado en los anteriores apartados. Todos tienen la característica que son *Open Source* menos *Gantter* que tiene un coste de 5€ al mes, aunque el primer mes es gratuito y se plantea solo utilizarlo únicamente en ese término, así pues todas las herramientas tienen coste 0.

La tabla 5 muestra los costes genéricos identificados anteriormente, su amortización y vida útil. Para calcular las amortizaciones, se ha calculado el coste por hora teniendo en cuenta que un año tiene 220 días hábiles y 8 horas laborables al día:

$$\text{Amortización} = \text{CosteDispositivo}/(\text{VidaÚtil} * 220 * 8) * \text{horas}$$

Para el despacho no hace falta calcular el coste de amortización ya que es un activo alquilado, el resto de activos, al tener un coste inicial 0, su correspondiente amortización acaba valorándose en 0€.

Elemento	Coste	Vida útil	Coste amortizado
Despacho	90€/mes	-	-
Ordenador	2.500€	5 años	92,89€
Software	0€	0	0€
Total	**2.860€/h**	-	**93€**

Tabla 5: *Costes genéricos.*

7.1.3. Contingencia

Como en todo proyecto, es importante añadir un sobrecoste para cubrir obstáculos e imprevistos. En este caso se ha decidido fijar un 15 % de sobrecoste. En la Tabla 6 se detalla la contingencia total del proyecto:

Tipo	Coste	Contingencia
Espacio	360€	54€
Hardware	92,89€	13,93€
Personal	12.847€	1.927,05€
Total	**13.300€**	**1.954€**

Tabla 6: *Tabla de contingencia del 15 % por tipo de gasto.*

7.1.4. Imprevistos

Por último, se calcula el coste de los obstáculos que puedan surgir durante el desarrollo del proyecto. Los imprevistos se presentan en la planificación temporal, a continuación, sólo se cuantifica el riesgo y el coste que pueden causar, en la Tabla 7 se detalla el coste.

- Fallo de dispositivo. En caso de que el ordenador falle será necesario comprar uno nuevo. Se estima un 1 % de riesgo ya que el ordenador y sus componentes son nuevos.
- Aumento tiempo de desarrollo, se añadirán 20 horas de desarrollo a la aplicación y 2 horas de *testing*. El coste total sería de 20 horas de programador y 2 horas de *tester*, por lo tanto, 352€, se cuantifica el riesgo en un 5 % ya que se hacen reuniones de forma periódica con los solicitantes de la aplicación.
- Errores en *Godot*: Tal y como se describe en la planificación temporal, en caso de que hubiera algún *bug*, se habría que buscar otra solución alternativa, por lo tanto se añadirían 25 horas de trabajo y 3 horas de *tester*, en total 448€. Se estima un riesgo relativamente bajo ya que dicha herramienta es utilizada por miles de usuarios.

Imprevisto	Coste	Riesgo	Coste total
Ordenador	2.500€	1 %	25€
Aumento tiempo desarrollo	352€	5 %	16,25€
Errores en Godot	448€	5 %	22,4€
Total	**3.300€**	-	**63,65€**

Tabla 7: *Tabla de sobrecostes añadido por imprevistos.*

7.1.5. Coste total

Una vez presentados todos los costes del proyecto, en la Tabla 8 se presenta el presupuesto final del trabajo. El coste total del proyecto es de 15.459€.

Tipo	**Coste**
Personal	12.847€
Espacio	360€
Hardware	92,89€
Contingencia	1.994,98€
Imprevistos	63,65€
Total	**15.359€**

Tabla 8: *Tabla del presupuesto final del proyecto.*

7.2. Control de gestión

Una vez definido el presupuesto inicial, se definen los mecanismos de control necesarios para evitar desviaciones, así como indicadores numéricos que ayuden al control. En las reuniones semanales, cada vez que se acabe una tarea, se actualizará el presupuesto con las horas reales y se comparará con las horas estimadas.

Para controlar los imprevistos, al finalizar una tarea en *Trello* también se apuntarán los gastos extra que se hayan producido en una hoja de *Excel*, y se compararán con la previsión de imprevistos y contingencia. De esta forma, rápidamente se podrá detectar cualquier desviación y predecir si es necesario recortar alguna tarea o aumentar el presupuesto.

A continuación se presenta los descriptores numéricos para el control:

- Desviación coste personal por tarea:
  (coste_estimado - coste_real) * horas_reales
- Desviación realización tareas:
  (horas_estimadas - horas_reales) * coste_real
- Desviación total en la realización de las tareas:
  (coste_estimado_total - coste_real_total)
- Desviación total de recursos (*hardware*, personal, espacio):
  (coste_estimado_total - coste_real_total)
- Desviación total coste de imprevistos:
  (coste_estimado_imprevistos - coste_real_imprevistos)
- Desviación total de horas:
  (horas_estimadas - horas_reales)

8. Sostenibilidad

8.1. Autoevaluación

Después de realizar la encuesta de sostenibilidad he visto que mis conocimientos relacionados con la materia son escasos. Algunas preguntas me han hecho dar cuenta de que los efectos colaterales de una solución relacionada con el mundo de la informática puede ser muy drásticos. También me he dado cuenta que el *hardware* que se ejecuta los programas requiere mucha energía y recursos para ser producido.

Otro aspecto importante es el social. En este ámbito si que personalmente conozco las ventajas de las herramientas colaborativas, y también que el *software* ha de ser ético, transparente y accesible, ya que su objetivo es ayudar a la gente.

8.2. Dimensión Económica

El coste estimado de este proyecto incluye únicamente los recursos necesarios para el correcto desarrollo y funcionamiento de la aplicación y también se han estudiado los posibles riesgos que hacen variar este presupuesto, por lo tanto se intenta no hacer un mal uso de dinero ni recursos.

8.3. Dimensión Ambiental

Respecto a la sostenibilidad ambiental, se ha tenido en cuenta de que el *software* no haga uso de recursos innecesarios o ineficientes, aunque está el impacto negativo latente a raíz de la electricidad consumida. No obstante, el proyecto ha sido desarrollado en un despacho ubicado en mi vivienda, así que no se provocará ningún impacto medioambiental por desplazamientos. A parte de la electricidad, no hay otro tipo de recurso que impacte negativamente el medio ambiente, ya que no hace falta ningún *hardware*, se puede desarrollar y utilizar con equipos ya amortizados y no se produce CO_2 durante la elaboración del proyecto. Se ha hecho hincapié en utilizar herramientas *Open Source* con tal de aprovechar la energía que se gastaría desarrollando funcionalidades ya existentes.

8.4. Dimensión Social

Personalmente, este proyecto puede enriquecerme de varias formas, no solamente con el conocimiento sobre el desarrollo de *software* y cómo aplicar las metodologías ágiles. También se aprenderá a cómo llevar sesiones de *testing* con los solicitantes de este proyecto para poder revisar funcionalidades y ver si se pueden replantear con el fin de que sean más intuitivas y utilizables. Y por último, y no por ello menos importante, este proyecto nace de la necesidad del mundo sanitario para poder diagnosticar y tratar la salud visual de las personas, y sobre todo de las personas de edad avanzada que forman parte del grupo de riesgo de la sociedad actual.

9. Bibliografía

[1] Federico. *¿Qué es la optometría deportiva?* Mar. de 2018. URL: https://www.federopticos.com/blog-optometria-deportiva/.

[2] Elisa Aribau. *El test de Snellen y la agudeza visual.* Mar. de 2018. URL: https://www.elisaribau.com/test-snellen-la-agudeza-visual/.

[3] *Dictionary of visual science ; edited by David Cline, Henry W. Hofstetter, John R. Griffin.* 1980.

[4] Lluïsa Quevedo et al. «A novel computer software for the evaluation of dynamic visual acuity». En: *Journal of Optometry* 5.3 (2012), págs. 131-138. ISSN: 1888-4296. DOI: https://doi.org/10.1016/j.optom.2012.05.003. URL: https://www.sciencedirect.com/science/article/pii/S188842961200057X.

[5] Administrator. *Optotipo Anillo disco palomar 'Optotipo Universal'.* URL: https://www.centrospalomar.com/index.php?option=com_content&view=article&id=246%3Aoptotipo-anillo-disco-palomar-qoptotipo-universalq-&Itemid=&lang=galego.

[6] *Trello ayuda a Los equipos a sacar el trabajo adelante.* URL: https://trello.com/es.

[7] *Where the world builds software.* URL: https://github.com/.

[8] *1 cloud-based project management software.* URL: https://www.gantter.com/.

[9] Unity Technologies. URL: https://unity.com/es.

[10] Godot Engine. *Free and open source 2D and 3D game engine.* URL: https://godotengine.org/.

[11] *Requirements test.* URL: https://www.systemrequirementslab.com/cyri/requirements/godot-engine/19806.

[12] *Overleaf, online latex editor.* URL: https://www.overleaf.com/.

[13] Sergio Luján Mora. *Herramientas para la Investigación.* URL: http://desarrolloweb.dlsi.ua.es/cursos/2015/herramientas-investigacion/que-es-latex.

[14] *ESGuiaHaysdelMercadoLaboral.* URL: https://www.hays.es/documents/63345/29167077/ESGuiaHaysdelMercadoLaboral2022.pdf.

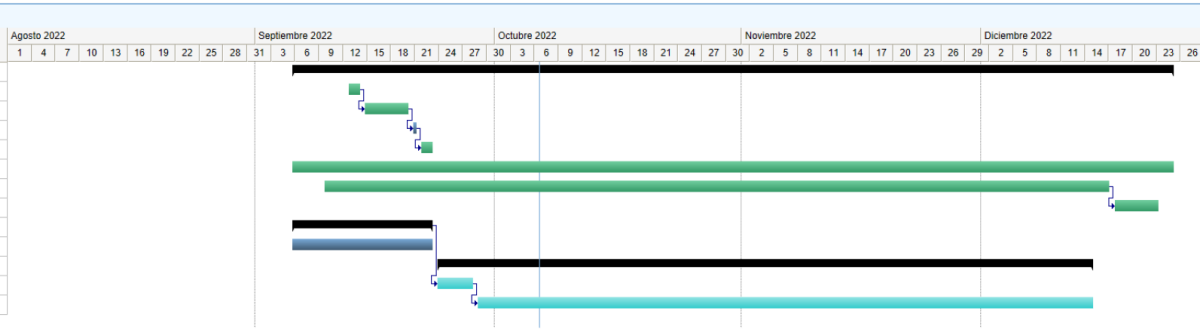
10. Anexos

Anexo A. Diagrama de Gantt

gantter™

Project Name **Sin Título**

	Nombre	Duración	Inicio	Fin	Predecesoras
1	**GP - Gestión del proyecto**	**80días?**	**09/05/2022**	**12/23/2022**	
2	GP.1 - Alcance	2días	09/12/2022	09/13/2022	
3	GP.2 - Planificación	4días	09/14/2022	09/19/2022	2
4	GP.3 - Prespuesto	1día?	09/20/2022	09/20/2022	3
5	GP.4 - Informe de sostenibilidad	2días	09/21/2022	09/22/2022	4
6	GP.5 - Reuniones	80días	09/05/2022	12/23/2022	
7	GP.6 - Documentación	70días	09/09/2022	12/15/2022	
8	GP.7 - Presentación	4días	12/16/2022	12/21/2022	7
9	**TP - Trabajo previo**	**14días**	**09/05/2022**	**09/22/2022**	
10	TP.1 - Aprendizaje	14días	09/05/2022	09/22/2022	
11	**DA - Desarrollo aplicación**	**58días**	**09/23/2022**	**12/13/2022**	
12	DA.1 - Diseño	3días	09/23/2022	09/27/2022	9
13	DA.1 - Implementación	55días	09/28/2022	12/13/2022	12



PNG Generated On: 6/10/2022, 18:24:08

Anexo A: *Diagrama de Gantt, producción propia.*