UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

20|21

Trabajo fin de grado

Desarrollo de una herramienta de cálculo para préstamos revolving

Jose Antonio Cebrián Diz

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Desarrollo de una herramienta de cálculo para préstamos revolving

Autor: Jose Antonio Cebrián Diz
Tutor: Manuel Pozo Muñoz
Ponente: Jose Ramón Dorronsoro Ibero

abril 2021

Jose Antonio Cebrián Diz
Desarrollo de una herramienta de cálculo para préstamos revolving

Jose Antonio Cebrián Diz
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A mi familia y amigos

Elige un trabajo que te guste
y no tendrás que trabajar ni un día de tu vida.

Confucio

Agradecimientos

En primer lugar me gustaría agradecer a mi familia el apoyo y la confianza que siempre han depositado en mí. Gracias a ello, he conseguido superar todos problemas que se me han planteado hasta ahora.

También me gustaría agradecer el apoyo a todos aquellos que empezaron siendo compañeros de carrera y se han convertido en grandes amigos.

Por último, me gustaría agradecer a algunos profesores el esfuerzo que han puesto impartiendo sus materias, despertando en mi un gran interés y manteniéndome motivado durante todo el aprendizaje.

Resumen

Los créditos son una operación financiera muy utilizada en la actualidad. Sin embargo, pocas son las personas que conocen realmente su funcionamiento y las implicaciones que puede tener contratarlos. Asimismo, los créditos revolving son una variante con unas características diferentes, en ellos la persona que recibe el dinero establece las condiciones de devolución del mismo (sujeto siempre a unos limites). Tras diferentes denuncias por malentendidos en las contrataciones de estos créditos, el Ministerio de Asuntos Económicos y Transformación Digital ha decidido intervenir, obligando a todas las entidades que ofrecen estos servicios a cumplir con una serie de requisitos.

Motivados por la necesidad de crear una herramienta que permita calcular estos créditos y satisfaga los requisitos legales, se ha desarrollado una aplicación web que cumpla con este mismo propósito. Mediante una interfaz amigable y una disposición de la información clara, la herramienta servirá para informar al usuario del desarrollo que seguirá el crédito que él mismo ha configurado.

Este proyecto se ha desarrollado utilizando diferentes tecnologías, que han dado como resultado un servicio web adaptado a todos los tipos de pantalla, capaz de generar informes dinámicos y mostrar gráficos con información representativa. Al mismo tiempo, todas estas caracteristicas se han desarrollado junto con un modelo de *single-page application*, lo que permite una mejor experiencia de usuario. Además, todo el servicio ha sido desplegado utilizando tecnología Cloud, aumentando así la disponibilidad del servicio frente a fallos externos a la propia herramienta.

Finalmente, hemos obtenido una aplicación que cumple con los requisitos legales, es fácil de usar y gracias a las decisiones de diseño constituye una herramienta robusta a la vez que escalable.

Palabras clave

Crédito revolving, calcular, aplicación web, SPA, tecnología Cloud, .NET Framework, Angular, Bootstrap, API, JWT.

Abstract

Nowadays, one of the most frequently used financial operations is credit. However, there is a small amount of people who really know how they work and what is involved in taking out one of them. Revolving credits are a special kind of credit, in which the borrower establishes the conditions of repayment (always under some limits). After several reports due to misunderstandings in the arrangement of these credits, the Ministry of Economic Affairs and Digital Transformation has decided to take action, forcing all entities that offer these services to meet a set of requirements.

Motivated by the need to create a tool to calculate these credits and meet the legal requirements, a web application has just been developed for this purpose, using a user-friendly interface and a clear information layout to inform users about the progress of the credit that they have set up.

This project has been developed using different technologies, which have resulted in a web service adapted to all screen types, able to generate dynamic reports and display charts with representative information. All of these are under a single-page application model, which allows a better user experience. In addition, the whole service has been deployed using Cloud technology, thus increasing the service's reliability against external failures to the tool itself.

Finally, we have obtained an application that meets the legal requirements, it is easy to use and thanks to design decisions,it is robust as well as scalable.

Keywords

Revolving credit, calculator, web application, Cloud based technology, SPA, .NET Framework, Angular, Bootstrap, API, JWT.

Índice

Listas

Lista de códigos

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

Es un hecho que las nuevas tecnologías están cada día más presentes en nuestra sociedad y esto no es producto del azar. Al ser unas herramientas que nos facilitan nuestras tareas diarias, hace que cada vez sean algo más imprescindible. Dentro de estas nuevas tecnologías, la informática y el desarrollo de software nos permite solucionar problemas que de otra manera sería muy difícil, ya sea mediante automatización de procesos lógicos/matemáticos, almacenamiento de grandes cantidades de datos, etc.

Sin embargo, también pueden dar solución a problemas que, a priori, no parecen tener relación. Dado que este es un campo muy amplio y flexible, se puede adaptar para dar solución a diversos problemas. Durante el desarrollo de este TFG se verá cómo se han adaptado diferentes tecnologías de la información para proveer un servicio que ayude a entender mejor al usuario final todo lo relacionado con un crédito, más concretamente con un crédito revolving. Normalmente pensamos que el hecho de pedir un crédito es simplemente solicitar un dinero que al cabo de un tiempo iremos devolviendo con unos intereses asociados. Pero hay otras variables que si conocemos y modificamos nos ayudarán a darnos cuenta de que las condiciones que nos oferta la entidad emisora del crédito no son tan beneficiosas como pueden parecer en un primer momento.

1.1. La importancia de los créditos revolving

En la actualidad existen diferentes tipos de préstamos en función de la necesidad del solicitante. Los más comunes son préstamos hipotecarios, préstamos personales y préstamos al consumo. Cada uno de ellos tiene sus propias características, entre las que podemos destacar entidades que los ofrecen, intereses que se aplican, motivos para la adquisición del préstamo, cantidad máxima prestada, etc.

Las líneas de crédito revolving son una alternativa a los préstamos personales convencionales, sobre todo cuando se requieren unas características específicas tales como la velocidad en la concesión del préstamo o flexibilidad para elegir la cuota mensual. Además, estas líneas de crédito no están exclusivamente reservadas a entidades bancarias, sino que también pueden ser proporcionadas por otras entidades financieras.

Por otra parte, es importante señalar la diferencia entre una línea de crédito revolving y una tarjeta de crédito, ya que son conceptos que se pueden llegar a confundir. En el caso de revolving, la cantidad solicitada se paga mensualmente en cuotas previamente acordadas, mientras que con la tarjeta de crédito se abona toda la cantidad prestada en un día pactado. Por este motivo, se suele relacionar más los créditos revolving con los préstamos personales y no con las tarjetas de crédito convencionales.

Es conveniente conocer este tipo de crédito para ocasiones en las que necesitamos flexibilidad en los pagos y liquidez de forma urgente. Se suelen utilizar para efectuar pagos extraordinarios o imprevistos. En cualquier caso, no son recomendables para alargar demasiado el tiempo de devolución de la deuda, ya que uno de los inconvenientes de estos créditos es que el tipo de interés es muy elevado.

1.2. Necesidad de nuevas soluciones

Las herramientas actuales que nos permiten calcular diferentes líneas de crédito revolving en función de unos datos de entrada son muy pocas. La más completa es la que ofrece el Banco de España que está disponible en su página web. No obstante, dicha herramienta solo cuenta con una modalidad de pago que es cuota fija (todos los meses se paga la misma cuota hasta que se amortiza toda la deuda). Sin embargo, en los últimos años las entidades que proporcionan estas líneas de crédito también ofertan otras modalidades de pago, tales como elegir el número de pagos o elegir qué porcentaje de la deuda restante quieres amortizar en cada cuota.

A pesar de que la información más importante del cálculo se muestra de manera clara y sencilla, cuando el usuario quiere ver los flujos de amortización o la evolución del préstamo de una manera más visual, tiene que estar navegando por el simulador. Esto puede dar como resultado que el usuario no entienda correctamente todos los datos en conjunto.

1.3. Mi propuesta

Debido a la poca oferta de este tipo de aplicaciones y que las opciones de cálculo que ofrecen son muy limitadas, mi propuesta surge con la intención de suplir las carencias de las herramientas existentes. En ella, se mostrarán todos los resultados de la manera más clara y sencilla posible, de forma que cualquier persona pueda usarla y entenderla. Además, incorporará la posibilidad de descargar un informe donde se mostrará tanto el cálculo realizado como los resultados obtenidos. De esta forma, daremos la posibilidad al usuario de imprimirlo o compararlo con otros informes generados previamente.

Asimismo, para que sea accesible y usable por cualquier persona, se ha optado por desarrollar la herramienta en forma de aplicación web. Esta decisión proporciona una serie de ventajas, tales como eliminar molestos procesos de instalación o reducir los requisitos de ejecución (solo se necesita un navegador y conexión a internet). También reduce las posibilidades de que la aplicación falle en comparación con una aplicación convencional, ya que no ocasiona problemas técnicos debidos al hardware u otras aplicaciones que estén ejecutándose en el sistema.

En resumen, mi propuesta es una herramienta web lo más accesible posible, donde se ofrecen diferentes posibilidades de cálculo y se muestran los resultados de la forma más clara, de forma que, al usuario le sea sencillo el uso de la aplicación y la compresión de los resultados.

2

Las finanzas de revolving

Durante el desarrollo de este capítulo vamos a explicar qué es un crédito revolving y todo lo relacionado con él. Para ello, nos centraremos en aspectos como la legislación que envuelve a estos créditos y los cálculos matemáticos que hay detrás de ellos.

2.1. De qué trata

Según el artículo “Crédito revolving” del periódico elEconomista [1], un crédito revolving se puede definir de la siguiente manera: "línea de crédito concedida por una entidad financiera a un cliente, con un límite establecido del que puede disponer durante un tiempo determinado". La principal diferencia con respecto a una línea de crédito convencional es que, en este caso, el usuario puede elegir la cuota mensual que quiere pagar hasta liquidar la deuda. Además, otra característica que lo diferencia es que, al pagar una cuota mensual, la parte proporcional a la amortización vuelve a estar disponible para ser utilizada otra vez como parte del crédito.

Para entenderlo mejor vamos a explicar un ejemplo sencillo (los cálculos no serán realistas, pero servirán para entender el funcionamiento). Suponemos que vamos a una entidad financiera y nos conceden una tarjeta revolving, con una cantidad máxima de 10.000 € y una cuota mensual de 50 €. Nos llega la tarjeta el mes de febrero, con lo cual, ya podemos empezar a usarla y decidimos gastar 200 € en un móvil nuevo. En ese mismo momento debemos 200 € + 10 € de intereses. A finales de febrero nos llega el recibo y pagamos los 50 € acordados, por tanto, nos quedan por pagar 160 €. Durante el mes de marzo decidimos usar la tarjeta para comprarnos unas zapatillas nuevas que cuestan 80 €, con lo cual, nuestra deuda aumenta a (160 € + 80 €) + 11 € de intereses. En los próximos meses se seguirá pagando 50 € de cuota hasta liquidar la deuda.

Cabe destacar que, en el ejemplo, el límite máximo fueron 10.000 € y que entre la primera compra (el móvil) y el primer pago, la cuantía máxima que podíamos gastar se redujo a 9.800 €. Sin embargo, después de realizar el primer pago, el límite volvió a ascender a 9.850 €. Por esta característica de renovación que mencionamos recibe el nombre de revolving (del inglés *“to revolve”*).

Por otra parte, la calculadora que se ha desarrollado no tiene en cuenta posteriores renovaciones del crédito, ya que se centra en la cantidad inicial de la que se va a hacer uso. Por lo que en el ejemplo correspondería únicamente a solicitar 200 €.

En la siguiente tabla se muestran las ventajas y desventajas de usar este tipo de créditos.

Ventajas	Desventajas
Cómodas cuotas	Altos tipos de interés
Pocos requisitos para adquirirlo	Posibles endeudamientos infinitos
Dinero disponible desde el inicio	El impago de cuotas implica pagar más intereses

Tabla 2.1: En esta tabla se resumen algunas ventajas y desventajas de usar este tipo de créditos. Fuente: elaboración propia usando [1] [2].

2.2. Procedimientos estándar de concesión

Para optar a este tipo de crédito solo necesitamos encontrar una entidad que lo ofrezca y rellenar un contrato en el que se especifican entre otros los siguientes datos: importe máximo a financiar, tipo de interés, cuota mensual. Como veremos a continuación, los requisitos para obtener este tipo de créditos son prácticamente nulos.

Antes de la estandarización de internet las formas de obtener este tipo de créditos eran muy pocas y la más común era acudir a un banco y solicitarlo allí. Sin embargo, había otras opciones para conseguirlo. Una de ellas consistía en recibir una carta con un formulario ya redactado, el cual, se rellenaba y mandaba en el mismo sobre a la entidad financiera para que posteriormente enviaran la tarjeta. En otras ocasiones, iban comerciales de estas entidades a las empresas y rellenaban el contrato solicitándo datos como el nombre, apellidos, DNI y cuenta bancaría a en la que domiciliar los recibos. En líneas generales, la contratación era poco garantista y la entidad financiera no se aseguraba lo suficiente de que el dinero pudiese ser devuelto o si estaba sometiendo al cliente a una operación de alto riesgo, llevándole en ocasiones asumir una cuota que no era acorde con sus ingresos.

A pesar del paso de los años, la tendencia de no analizar lo suficiente la solvencia del solicitante continua, produciendo muchas veces un endeudamiento mayor. A partir de una sentencia del Tribunal Supremo “STS 600/2020” [3] de marzo de 2020 se empieza a ver la necesidad de alertar a los consumidores de todo lo que implica este tipo de créditos antes de contratarlos. Por ello, el Ministerio de Economía publicó una orden en el BOE “ETD/699/2020, de 24 de julio” [4] donde obliga a las entidades, entre otras cosas, a analizar la solvencia de los solicitantes y a mostrarles información detallada antes de firmar el contrato. Dicha información ha de ir acompañada de ejemplos representativos que permitan al cliente entender mejor qué están contratando. Esta nueva orden del BOE entrará en vigor a lo largo del año 2021.

Todos los cambios que introduce esta nueva orden en el BOE serán expuestos en el siguiente apartado. Además, explicaremos la motivación de estos, ya que en ocasiones se llegaban a producir endeudamientos perpetuos, los cuales, surgían por la elección de una cuota mensual demasiado baja respecto a la cantidad solicitada.

2.3. Legislación

Como hemos comentado en el apartado anterior, una mala elección de la cuota mensual puede dar lugar a endeudamientos infinitos o a pagar unos intereses muy elevados. Como muy bien se explica en el siguiente artículo de el periódico ABC [2], cuando el consumidor elige una cuota baja con respecto a la cantidad solicitada, en cada cuota deja parte de los intereses sin pagar por lo que prácticamente no amortiza capital. Además esa parte de los intereses que deja sin pagar pasan a ser parte de la deuda pendiente, por lo que la deuda crecería indefinidamente. En caso de elegir una cuota demasiado baja (aún cubriendo los intereses en cada cuota) podríamos encontrarnos con situaciones en las que los intereses totales que vamos a pagar son prácticamente iguales a la cantidad que habíamos solicitado. En el artículo se explica con el siguiente ejemplo: con una deuda de 1000 € a un tipo de interés del 25,61 %, si elegimos una cuota de 25 € tardaríamos 6 años y 10 meses en amortizar el crédito. Todo ello pagando de intereses 1.031,90 €, prácticamente la misma cantidad que la solicitada aun habiendo acordado un tipo de interés del 25,6 %

Una de las muchas denuncias de la gente en contra de estos intereses tan altos llegó a manos del Tribunal Supremo en marzo de 2020. La sentencia del Tribunal [3] anuló el préstamo de una clienta que había contratado una tarjeta revolving con unos intereses del 27 % TAE. La defensa de la prestataria se fundamentaba en la "Ley de 23 de julio de 1908 sobre la nulidad de contratos de préstamos usurarios" [5]. Además el Tribunal aprovechó para recordar que, en una sentencia de 2015 relacionada con estos créditos, declaró que las entidades que ofrecen estos servicios no pueden cobrar intereses tan elevados basandose en que muchos de sus clientes no serán capaces de pagarlos completamente. Más aún cuando son ellos mismos los que conceden estos créditos sin analizar la capacidad financiera de los prestatarios.

Ante esta última sentencia del Tribunal y los diferentes problemas que causaban a los usuarios este tipo de créditos el Ministerio de Asuntos Económicos y Transformación Digital decidió intervenir y publicó una Orden en el BOE [4] con la principal intención de proteger al usuario frente al desconocimiento de estos tipos de créditos tan “cómodos” a primera vista.

Algunos de los aspectos más relevantes que recoge esta nueva orden son:

- **Asegurar la solvencia de los clientes**: Se realizara un análisis profundo sobre la capacidad financiera del prestatario. Además, se aconsejarán otras cuotas con el fin de que el usuario pueda hacerse cargo todas las cuotas durante la vida del crédito de manera que no aumente

el endeudamiento.

- **Ofrecer información precontractual con ejemplos representativos**: Esta información tiene que ser lo más clara, de forma que al usuario le sea lo más sencillo posible entender cómo se desarrollara el pago del crédito.
- **Ofrecer información periódica al cliente** con al menos la siguiente información:
  - Importe solicitado, tipo de deudor, modalidad de pago elegida, fecha de finalización y cuantía total que deberá pagar (diferenciando la parte proporcional a los intereses y la parte proporcional a la deuda).
  - Escenarios de posible ahorro aumentando la cuota mensual en un 20 %, 50 % y 100 % cuando la cuota elegida no supere el 25 % del crédito solicitado.
  - Importe de la cuota que permite liquidar el crédito en un periodo de 12 meses.

Sin embargo, en dicha orden, no se hace ningún tipo de referencia a limitar el tipo de interés. Desde el Ministerio lo justifican diciendo que podría ser una medida contraproducente, ya que en el caso de limitarlo de forma alcista, las entidades aplicarían el tipo más alto posible y el resto de las ganancias las obtendrían aumentando otro tipo de costes. En caso contrario, si lo limitan bajando el actual, tendríamos como resultado que muchas familias no podrían optar a estos créditos ya que las entidades aumentarían más los requisitos con el fin de asegurarse el pago íntegro de la deuda y los intereses.

Tanto los aspectos mencionados anteriormente como otros menos relevantes que también están presentes en esta nueva orden entrarán en vigor a lo largo del año 2021.

Cabe destacar que, todas estas consideraciones recogidas en el BOE, se han tenido en cuenta a la hora de desarrollar la aplicación con el fin de que esta cumpla con la legislación vigente y la que entrará en vigor en 2021.

2.4. Formulación matemática de los cálculos

2.4.1. Nomenclatura y bases de cálculo

Para poder explicar de forma sencilla los cálculos realizados para este tipo de créditos, es necesario entender que es un flujo de caja financiero. Como muy bien se expone en este blog [6] “El flujo de caja financiero se define como la circulación de efectivo que muestra las entradas y salidas de capital de una empresa fruto de su actividad económica.” En nuestro caso, cada fila del flujo de caja representará la fecha en la que deberá pagar la próxima cuota. Así como el estado general de la deuda hasta esa fecha.

Otro factor relevante es la base de cálculo que vamos a utilizar. Algunas de las más utilizadas son

las siguientes:

- **30/360**: Para el cálculo de intereses supone que todos los meses tiene 30 días (año comercial) y utiliza una base de cálculo de 360 días.
- **Actual/360**: Todos los meses se computan por los días reales que tienen y utiliza una base de cálculo de 360 días.
- **Actual/365**: Todos los meses se computan por los días reales que tienen y utiliza una base de cálculo de 365 días.
- **Actual/Actual**: Tanto los meses como los años se computan por los días reales que tienen.
- **30/365**: Para el cálculo de intereses supone que todos los meses tiene 30 días (año comercial) y utiliza una base de cálculo de 365 días.

Para hacer la explicación de los cálculos lo más sencilla posible voy a explicar algunos términos y notaciones a las que haré referencia en el resto de éste capítulo.

- **Cuota (i)**: Cantidad total que tendrá que pagará el cliente en el mes i.
- **Interes (i)**: Cantidad de dinero asociada a intereses que pagará el cliente en el mes i.
- **Amortización (i)**: Cantidad de dinero que amortizará el cliente en el mes i.
- **Fecha (i)**: Fecha del pago de la cuota en el mes i. En caso de que la fecha no sea un día hábil, utilizando la convención PBD (Previous business day) el día de pago pasará a ser el día hábil anterior.
- **Pendiente (i)**: Cantidad de dinero que le queda por amortizar después de pagar la cuota del mes i.
- **Dif_d(i)**: Cantidad de días entre **Fecha(i)** y **Fecha(i-1)** teniendo en cuenta la base de cálculo.
- **Base**: Cantidad de días que tiene el año según la base de cálculo elegida.
- **Ti**: Tipo de interés aplicado.
- **Ft**: Cantidad neta que percibe la entidad financiera en el mes i.

Por último, durante la explicación vamos a suponer que el mes 0 es el inicial y a partir del mes número 1 se empiezan a pagar las cuotas.

2.4.2. Cálculo con cuota fija

En este tipo de cálculo el cliente fija cuota mensual que pagará hasta liquidar la deuda, este valor será representado con C_e.

La primera fila del Flujo de caja quedaría de la siguiente manera:

Fecha(0) = Fecha en la que se solicita el crédito.

Cuota(0) = Interés(0) = Amortización(0) = 0 €.

Pendiente(0)= Cantidad solicitada a la entidad financiera.

Formulación general:

(2.1)

(2.2)

$$Cuota(i) = \text{mín}\{C_e, Interés(i) + Pendiente(i-1)\} \tag{2.3}$$

$$Amortización(i) = Cuota(i) - Interés(i) \tag{2.4}$$

$$Pendiente(i) = Pendiente(i-1) - Cuota(i) \tag{2.5}$$

Ejemplo de flujo resultante:

Datos iniciales: base de cálculo = 30/360 , T_i = 16 %, cantidad solicitada = 100 € , C_e = 15 €.

Fecha	Cuota	Interés	Amortización	Pendiente
08/12/2020	0,00 €	0,00 €	0,00 €	100,00 €
08/01/2021	15,00 €	1,33 €	13,67 €	86,33 €
08/02/2021	15,00 €	1,15 €	13,85 €	72,48 €
08/03/2021	15,00 €	0,97 €	14,03 €	58,45 €
08/04/2021	15,00 €	0,78 €	14,22 €	44,23 €
07/05/2021	15,00 €	0,57 €	14,43 €	29,80 €
08/06/2021	15,00 €	0,41 €	14,59 €	15,21 €
08/07/2021	15,00 €	0,20 €	14,80 €	0,41 €
06/08/2021	0,42 €	0,01 €	0,41 €	0,00 €

Tabla 2.2: Ejemplo de un flujo de caja para el cálculo de cuota fija. Fuente: elaboración propia.

2.4.3. Cálculo con N plazos

En esta modalidad de cálculo el cliente elige cuantas cuotas quiere pagar durante la vida del crédito. En este caso no existen unas fórmulas explicitas para realizar la estimación correspondiente a N meses. Por ello, se hace uso del cálculo de cuota fija, variando esta hasta conseguir el número de plazos deseados. Esto se puede realizar de muchas maneras computacionalmente hablando. Sin embargo, el propósito de este capítulo no es explicar cómo ha sido implementado, por lo que se desarrollará más este tema en próximos apartados.

Ejemplo de flujo resultante:

Datos iniciales: base de cálculo = 30/360 , T_i = 16 %, cantidad solicitada = 3000 € , N = 8 plazos.

Fecha	Cuota	Interés	Amortización	Pendiente
11/12/2020	0,00 €	0,00 €	0,00 €	3.000,00 €
12/01/2021	430,56 €	41,33 €	389,22 €	2.610,78 €
12/02/2021	430,56 €	34,81 €	395,74 €	2.215,04 €
12/03/2021	430,56 €	29,53 €	401,02 €	1.814,02 €
12/04/2021	430,56 €	24,19 €	406,36 €	1.407,66 €
12/05/2021	430,56 €	18,77 €	411,78 €	995,88 €
11/06/2021	430,56 €	12,84 €	417,72 €	578,16 €
12/07/2021	430,56 €	7,97 €	422,58 €	155,58 €
12/08/2021	157,65 €	2,07 €	155,58 €	0,00 €

Tabla 2.3: Ejemplo de un flujo de caja para el cálculo de N plazos. Fuente: elaboración propia.

2.4.4. Cálculo con cuota variable

En este tipo de cálculo el cliente elige el porcentaje que quiere amortizar de la deuda pendiente en cada cuota. Este porcentaje será representado con P_e. Cabe destacar que, como el usuario elige un porcentaje, cuando la deuda pendiente es pequeña tendríamos amortizaciones cada vez más pequeñas. Para evitar que el cálculo se haga infinito, fijamos una amortización mínima de 15 €. Esta amortización la designaremos con A_m.

La primera fila del Flujo de caja quedaría igual que en el cálculo anterior.

Formulación general:

$$\boxed{Fecha(i) = Fecha(i-1) + 1\ mes} \tag{2.6}$$

$$Amortización(i) = \text{máx}\{P_e * Pendiente(i-1),\ \text{mín}\{Pendiente(i-1),\ A_m\}\} \tag{2.7}$$

$$Interés(i) = \frac{Pendiente(i-1) * T_i * Dif\_d(i)}{Base} \tag{2.8}$$

$$Cuota(i) = Amortización(i) + Interés(i) \tag{2.9}$$

$$Pendiente(i) = Pendiente(i-1) - Cuota(i) \tag{2.10}$$

Ejemplo de flujo resultante:

Datos iniciales: base de cálculo = 30/360 , T_i = 16 %, cantidad solicitada = 100€ , P_e = 30 %.

Fecha	Cuota	Interés	Amortización	Pendiente
11/12/2020	0,00 €	0,00 €	0,00 €	100,00€
12/01/2021	31,38 €	1,38 €	30,00 €	70,00€
12/02/2021	21,93 €	0,93 €	21,00 €	49,00€
12/03/2021	15,65 €	0,65 €	15,00 €	34,00€
12/04/2021	15,45 €	0,45 €	15,00 €	19,00€
12/05/2021	15,25 €	0,25 €	15,00 €	4,00€
11/06/2021	4,05 €	0,05 €	4,00 €	0,00€

Tabla 2.4: Ejemplo de un flujo de caja para el cálculo de cuota variable. Fuente: elaboración propia.

2.4.5. Cálculo de la TAE

Un factor importante a la hora elegir un crédito u otro es la TAE. Como muy bien se indica en [7] y [8] la Tasa Anual Equivalente o Tasa Anual Efectiva es el resultado de una operación matemática, que utiliza el tipo de interés nominal y la frecuencia de los pagos. Con este resultado podemos conocer de forma más precisa el coste anual del crédito.

Sin embargo, la fórmula más común de la TAE no tiene en cuenta otros tipos de gastos, como por ejemplo, gastos de apertura, comisiones por sacar dinero del cajero, etc. Para tener todas estas cosas en cuenta, vamos a calcular el VAN (valor actual neto) el cual hace uso de la TAE. Cuando el VAN = 0, quiere decir que ninguna de las partes (entidad financiera o cliente) se han quedado con dinero que no les pertenece. Por lo tanto, la amortización del crédito se ha realizado correctamente. En otras

palabras, vamos a ir variando la TAE hasta conseguir que el VAN sea igual a 0.

Para explicar mejor el cálculo vamos a puntualizar un cálculo intermedio que estaría implícito, se hará referencia a él con $I(i)$. Por último, como este cálculo se realiza después de la caja de flujo, llamaremos N al número de cuotas que tendrá que pagar el cliente .

Inicialización de valores: $F_t(0)$ = -Pendiente(0), $I(0)$=0 .

$$F_t(i) = Cuota(i) \tag{2.11}$$

$$I(i) = (1 + TAE)^{(Dif\_d(i)/Base)} \tag{2.12}$$

$$VAN = F_t(0) + \sum_{i=1}^{N} \frac{F_t(i)}{(\prod_{x=1}^{i} I(i))} \tag{2.13}$$

Ejemplo de cálculo resultante:

Datos iniciales: base de cálculo = Actual/360 , T_i = 16 %, cantidad solicitada = 100 € , C_e = 15 €, TAE = 17,2550249... % .

Fecha	Cuota	Dif_d	Ft	I
24/01/2020	0 €	0	-100,00 €	0
25/02/2020	15,00 €	32	15,00 €	1,0142500
25/03/2020	15,00 €	29	15,00 €	1,0129055
24/04/2020	15,00 €	30	15,00 €	1,0133535
25/05/2020	15,00 €	31	15,00 €	1,0138017
25/06/2020	15,00 €	31	15,00 €	1,0138017
24/07/2020	15,00 €	29	15,00 €	1,0129055
25/08/2020	15,00 €	32	15,00 €	1,0142500
25/09/2020	0,54 €	31	0,54 €	1,0138017
			VAN	**0**

Tabla 2.5: Ejemplo de cálculo del VAN a partir de la TAE. Fuente: elaboración propia.

2.5. Necesidades de mejora con este TFG

Las calculadoras de créditos revolving que hay en la actualidad no implementan todas las características que ofrecen las entidades financieras. Tampoco cumplen los aspectos legales mencionados anteriormente. Por ello, con este TFG se busca cubrir todas estas necesidades dando como resultado una herramienta completa y útil, cumpliendo los requisitos legales que enuncia el BOE en relación con estos créditos.

Dentro de estos aspectos legales, podemos destacar los siguientes:

- **Información precontractual con ejemplos representativos:** la propia herramienta sirve para generar ejemplos representativos, los cuales, irán acompañados de gráficos y cuadros de amortización para que el usuario entienda mejor cómo se desarrollará el crédito.

- **Diferenciación clara de los intereses y de la amortización total:** esta diferenciación se podrá ver en el resultado final del cálculo, en cada uno de los flujos del cuadro de amortización y en los gráficos. De manera que el usuario será informado de manera explícita y en repetidas ocasiones de la cantidad de intereses que deberá pagar.

- **Escenarios de posible ahorro:** como determina el BOE, cuando la cuota mensual seleccionada por usuario sea menor del 25 % de la cantidad solicitada, se le presentarán a modo de recomendación diferentes escenarios de posibles ahorros, aumentando la cuota seleccionada en un 20 %, 50 % y 100 %. Adicionalmente se le indicará que cuota debería pagar para finalizar la deuda en un año.

- **Escenario de posible endeudamiento perpetuo:** cuando el usuario elige una cuota mensual que da como resultado un endeudamiento perpetuo es informado de manera inmediata imposibilitando además el cálculo del crédito.

Con el fin de adaptarse a todas las modalidades de pago que ofrecen las entidades, se han implementado 3 variantes del cálculo haciendo posible que el usuario interactúe con ellas y decida cuál de ellas se acerca más a sus necesidades. Por ejemplo, en el caso de cuota variable, los primeros pagos son más elevados y los últimos son más bajos.

3

Bases tecnológicas

Una vez se hemos definido los requisitos funcionales de la aplicación, el siguiente paso es decidir qué tecnologías vamos a usar. Este es un paso muy importante y dependerá de los requisitos no funcionales de la aplicación. En este caso, la aplicación será una herramienta web, por lo que a lo largo de este capítulo nos centraremos en explicar qué tecnologías, patrones y arquitecturas se van a utilizar.

3.1. Cloud

Desde la aparición de internet cada vez más y más empresas están optando por digitalizarse, pasando toda la documentación a un formato digital. En muchas ocasiones resulta interesante para la empresa/entidad que estos datos que han sido digitalizados sean accesibles para los trabajadores, es decir, que la información este centralizada. Para ello, normalmente se opta por mantener la información en unos servidores centrales que sean accesibles mediante la red. Asimismo, también puede resultar interesante para las empresas utilizar estos servidores para ejecutar tareas pesadas de forma centralizada.

Sin embargo, el hecho de depender de unos servidores puede acarrear una serie de inconvenientes. Por ejemplo, disponer de un técnico las 24 horas del día que pueda solucionar cualquier incidencia relacionada con la disponibilidad de los mismos. En el peor de los casos, cuando el servidor deja de proporcionar servicio por cuestiones relacionadas con el hardware, la entidad tendría que parar su actividad, lo que podría traducirse en pérdidas económicas. Para que esto no ocurra es necesario que los equipos y la instalación donde están localizados cumpla con unos requisitos de disponibilidad que no son siempre fáciles de satisfacer.

Por lo tanto, con el fin de solucionar este y otros problemas, surge la idea de *“Cloud computing”* (Computación en la nube), un modelo de negocio en el que se alquilan recursos informáticos. Este se basa en suscripciones mensuales o anuales, donde únicamente se tendrá acceso a los recursos mientras dure la misma. Este tipo de negocio lo vemos cada día más implantado en la sociedad ya que facilita mucho el uso de los servicios en forma de software sin tener que preocuparnos por otros

problemas que puedan surgir. En el caso de proporcionar servicios centralizados, el hacer uso del Cloud nos libera de mantener la disponibilidad de los servicios que queremos proporcionar, ya que esto es responsabilidad de la empresa que nos oferta el alquiler de dichos recursos.

A pesar de que la frase “alquilar recursos informáticos” es un poco difícil de entender, en el siguiente apartado explicaremos de forma detallada los servicios más comunes que se ofrecen. Además, expondremos algunos ejemplos para dejarlo lo más claro posible.

3.1.1. Tipos de servicios que se ofertan

El modelo de negocio Cloud abarca tres tipos de servicios, los cuales son:

- **SAAS (Software as a service):** en este tipo de servicio, las empresas ofrecen un software y una infraestructura ya creada. De esta forma el cliente lo único que tiene que hacer es usar dicho software sin preocuparse de instalaciones o configuraciones complejas. Es curioso ver cómo cada vez más y más empresas están optando por este modelo de negocio. Cuando surgió la venta de software todas las empresas comercializaban sus productos en un Floppy o en un CD listos para empezar a usarse. El problema de este tipo de distribución llegó rápidamente con la piratería, de forma que los desarrolladores optaron por crear licencias para poder usar sus programas. Sin embargo, estas licencias tenían un problema, se generaban siguiendo un patrón secreto que tarde o temprano se acababan descubriendo y la seguridad de sus productos se veía comprometida. Por lo tanto, la forma más segura de proteger el software actualmente es vendiéndolo en forma de servicio, haciendo que los usuarios tengan que suscribirse para poder usarlo. Uno de los ejemplos más recientes de empresas que se han pasado a este modelo es Adobe con su software “Adobe Photoshop”.

Algunos de los servicios SAAS más utilizados son Netflix, Dropbox, Gmail, GDrive y Spotify.

- **IAAS (Infrastructure as a service):** esta modalidad de Cloud se basa en ofrecer un servicio de infraestructura informática bajo demanda, es decir, alquilar potencia de cómputo según necesite el cliente. Gracias a esa capacidad de cómputo y a la infraestructura proporcionada el cliente es capaz de ejecutar el software que desee de forma remota. Dicha capacidad de cómputo se puede medir en número de núcleos, tamaños de memoria primaria y secundaria, etc. En este tipo de servicio el cliente solo tiene que encargarse de que el software que va a ejecutar funcione correctamente y todo lo relacionado con los problemas de hardware que pudieran surgir (reinicios, apagones, sobrecalentamientos) son responsabilidad de la empresa que oferta el servicio Cloud. Este tipo de modalidad surge de lo que se conocía anteriormente como *“Hosting”*, donde se alquilaba un equipo entero para ejecutar un software. Gracias a los grandes *data centers* equipados con las últimas tecnologías y unas capas de abstracción, se ha conseguido que con unos pocos “clicks” se puedan crear máquinas virtuales en los *data centers* y se alojen en los equipos libres o que dispongan de los recursos solicitados. De esta manera, al no alquilar equipos completos (pagamos solo lo que necesitamos) se consigue un mejor

aprovechamiento de los equipos y un precio mas competitivo. Esto, da como resultado la posibilidad de tener servidores de cualquier tamaño y potencia, haciendo transparente para el usuario la creación y alojamiento de la máquina virtual. Cabe destacar que en esta modalidad, el usuario se tiene que encargar de configurar el entorno de despliegue del software. También tendrá que realizar las configuraciones que requiera el sistema operativo dentro de la máquina virtual.

Algunos de los servicios IAAS más utilizados son Azure (Microsoft), AWS (Amazon) y Google Cloud.

- **PAAS (Platform as a service):** este tipo de servicio se encuentra entre los dos mencionados anteriormente. El público objetivo son los desarrolladores software que necesitan desplegar aplicaciones en la nube de forma rápida, sin tener que estar configurando parámetros del sistema operativo o de despliegue de la aplicación. La propia empresa que oferta este modelo se encarga de tener actualizados los equipos, de forma que el usuario, no se tenga que preocupar por ello. En contraposición a la rapidez de despliegue y configuración, nos encontramos con poca flexibilidad en el desarrollo de las aplicaciones. En estos casos, tanto los lenguajes de programación como las bases de datos que soportan estos servicios, son limitadas y tendremos que adaptarnos desarrollando software compatible que pueda ser ejecutado en esas máquinas.

Algunos ejemplos de servicios PAAS más utilizados son Elastic Beanstalk (Amazon), App Engine (Google), Azure (Microsoft), Cloud (IBM).

3.1.2. Principales proveedores de servicios Cloud

Cada vez más empresas están optando por ofrecer este tipo de servicios. Como podemos ver las que lideran este sector son Amazon, Microsoft y Google. Sin embargo hay otras que, aunque no estén liderandolo, están más especializadas en algunas regiones geográficas o no ofrecen todos los tipos de servicios explicados anteriormente. A continuación se van a describir algunas características de los proveedores Amazon, Microsoft y Google.

- **Amazon Web Services:** Amazon cuenta con una infraestructura muy grande y por ello es capaz de tener diferentes *data centers* en EE. UU., Bahrein, Canadá, Francia, Alemania, Irlanda, Reino Unido, Australia, India, Italia, Singapur, Sudáfrica, Corea del Sur, Suecia, Brasil y Hong Kong. Por otra parte, AWS es el proveedor más fuerte en la mayoría de los casos. Además, su plan para el futuro es dominar el sector de los servicios Cloud consiguiendo cada vez mayor cuota de mercado.

Una de sus principales ventajas frente a sus competidores son las habilidades y los recursos para proporcionar las soluciones que mejor se adaptan a sus clientes. En caso de que fuese necesario, Amazon tendría la posibilidad de diseñar desde el silicio de sus procesadores hasta los sistemas operativos que utilizan, todo ello con el fin de ofrecer un mejor servicio.

- **Microsoft:** siendo uno de los líderes de este sector, cuenta con *data centers* en EE.UU., Reino Unido, Alemania, Australia, India, Noruega, Emiratos Árabes Unidos, Japón, Corea y Sudáfrica entre otros. Actualmente, lidera los proveedores de nubes a gran escala en términos de cuota de mercado en el segmento de desarrollo de aplicaciones PaaS. Esto se debe, en parte, a su conjunto de herramientas entre las que se encuentra Azure DevOps y Github.

El servicio Cloud denominado Microsoft Azure cuenta con características extra para el software desarrollado bajo el entorno de Microsoft, permitiendo por ejemplo, desplegar aplicaciones desde su "IDE" llamado Visual Studio. No obstante, más adelante se explicarán más características que ofrece este servicio.

- **Google:** por su parte, la empresa fundada por Larry Page y Serguéi Brin tiene sus principales *data centers* en Japón y EE.UU. Sin embargo, también está presente en países como Bélgica, Singapur, Finlandia, Alemania, Países Bajos, Reino Unido, India, Australia, Brasil, Hong Kong y Taiwán. Durante los últimos años ha estado trabajando en sus puntos débiles con el fin de poder ofrecer un buen servicio y competir con sus rivales directos (AWS y Azure). Además, planea realizar acuerdos con empresas de telecomunicaciones para mejorar aún más su servicio.

Por otra parte, las contribuciones de software libre de Google como Kubernetes y TensorFlow han revolucionado el mercado y han cambiado el curso del desarrollo software de las empresas. Con ello, Google ha conseguido acercarse al público del código libre. La intención con ello es que empiecen a usar su plataforma "Google Cloud Platform" para futuros proyectos.

Por último, a modo de resumen, podemos ver el estado de los principales proveedores de servicios Cloud según la clasificación realizada en el artículo publicado por la compañía Gartner.

Figura 3.1: Estado de los proveedores de servicios Cloud en 2020. Fuente: [9].

3.1.3. Azure

La herramienta de cálculo de créditos será desplegada en el entorno Cloud de Azure. Por tanto, vamos a explicar alguna de las herramientas que brinda Microsoft para los desarrolladores. Dentro del gran abanico de servicios que ofrece Azure, nos vamos a centrar en Azure DevOps. Con esta herramienta Microsoft pretende estandarizar una nueva metodología de trabajo que ellos denominan "Cultura Devops". Para ello, mezclan conceptos de metodologías ágiles con métodos de integración e implementación continua.

Algunas de sus herramientas principales son:

- **Azure Boards**: herramienta que facilita el uso de metodologías ágiles mediante paneles kanban. En estos paneles se colocan tarjetas con las tareas a realizar y posteriormente, según se vayan completando dichas tareas, se cambiará el estado de las tarjetas.
- **Azure Pipelines**: esta herramienta se encarga de los métodos de integración e implementación continua. Con una configuración previa es capaz de tomar la aplicación desde el servicio de control de versiones, compilarla y desplegarla.
- **Azure Repos:** servicio de control de versiones propiedad de Microsoft. Al estar basado en Git implementa todas las características que tendríamos en cualquier otro servicio similar. Además, junto con GitHub es compatible con Azure Pipelines.

Por último, vamos a ver de forma más la clara arquitectura de soluciones DevOps que se utilizará durante el desarrollo de este proyecto.

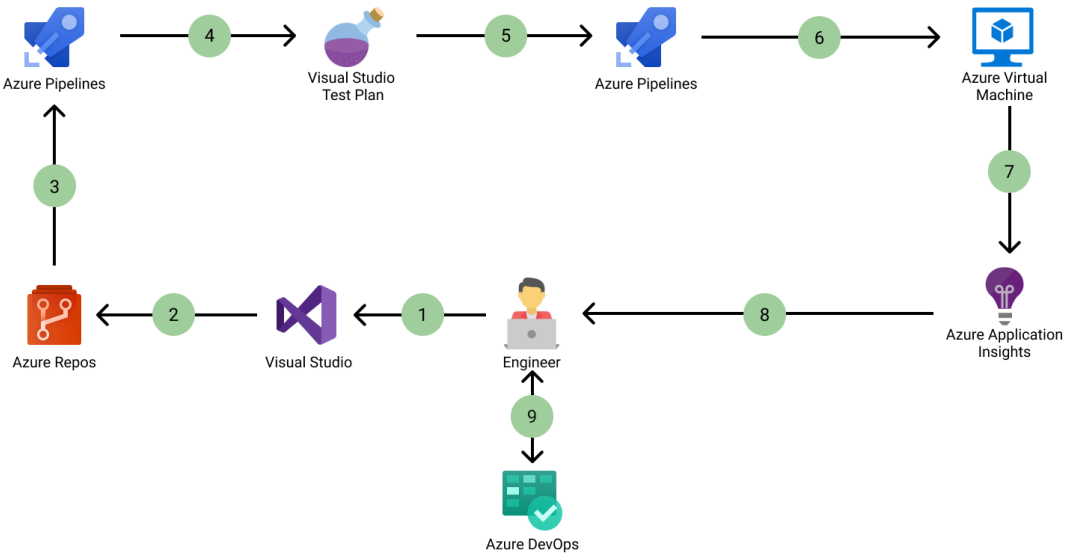

Figura 3.2: Arquitectura DevOps que se utilizará en el desarrollo de la herramienta web. Fuente: elaboración propia usando [10].

3.2. Aplicación web

Cada vez son más y más las herramientas web que utilizamos gracias, en parte, a la variedad de beneficios que ofrecen frente a otros tipos de aplicaciones. Sin embargo, pocas son las personas que entienden cómo funcionan realmente estas tecnologías; por ello vamos a explicar cómo funcionan de una manera sencilla. Empezaremos familiarizándonos con el vocabulario que se usa en estos contextos.

- **Cliente**: se denomina cliente al software que va a solicitar recursos. Para simplificar más las cosas y con el fin de ejemplificarlo mejor supondremos que el cliente es un navegador que se ejecuta en un ordenador convencional.
- **Servidor**: se denomina servidor al software que ofrece recursos al cliente. Estos recursos pueden ser imágenes, vídeos, procesamiento de cálculos, etc. Normalmente, estos dispositivos cuentan con unas prestaciones muy elevadas, ya que están enfocados a servir recursos a muchos clientes de forma simultánea.
- **Internet:** se denomina internet, a la infraestructura que hace posible la comunicación entre dispositivos a través de las redes de comunicación. Internet hace referencia tanto a la parte "física" de la red como también a la parte más "lógica", que hace posible que las comunicaciones sigan un orden y no sean un caos.

3.2.1. Arquitectura cliente-servidor

Este tipo de arquitectura es el estándar en las aplicaciones web. Define de forma muy estricta e invariable qué software actúa como cliente y qué software como servidor. Aquel programa que inicie la comunicación recibirá automáticamente el rol de cliente, mientras que el programa que recibe la comunicación asumirá el rol de servidor. Estos roles son asignados y no podrán cambiarse en ningún momento; si esto llegara a pasar no estaríamos delante de una arquitectura cliente-servidor.

Asimismo, existen otras características que definen esta arquitectura, entre las cuales podemos encontrar:

- **Seguridad**: en esta arquitectura el cliente que se comunica con el servidor no puede saber quién más está haciendo lo mismo. Desde el punto de vista del cliente el resto de las conexiones que mantenga el servidor serán transparentes.

- **Escalabilidad**: las máquinas que ejecutan el cliente y el servidor pueden aumentar sus prestaciones de forma independiente. Estos cambios no tienen que ser notificados y no afectarán nunca de forma negativa al servicio. Además, en cualquier momento se pueden incorporar clientes o servidores a la red, lo que facilita el uso de estos servicios de forma simultánea.

- **Control de la comunicación**: únicamente el cliente podrá iniciar la comunicación o solicitar recursos. El servidor, por su parte, solo podrá responder a las solicitudes del cliente, pero en ningún caso podrá solicitarle ninguna información. En caso de que el servidor obtenga información o recursos, será porque el cliente se lo ha comunicado previamente.

Una vez explicada la arquitectura vamos a exponer un ejemplo donde esté presente. Supongamos que queremos buscar algo en Google, para lo que utilizaremos el navegador del ordenador. En el momento en el que introducimos la dirección de la página web de Google nos convertimos automáticamente en los clientes de la arquitectura. Esto es debido a que le estamos solicitando a Google el contenido de su página web. Por otra parte, al recibir una solicitud, Google está actuando como servidor y tendrá que generar la página que le estamos solicitando para posteriormente enviarla de vuelta. De la misma forma, cuando vamos a realizar la búsqueda en Google rellenamos un cuadro de texto con las palabras que queremos buscar y le solicitamos al servidor que las busque por nosotros. El servidor realizará las búsquedas necesarias y devolverá la información de vuelta al cliente (en este caso el navegador) para que pueda ser visualizada.

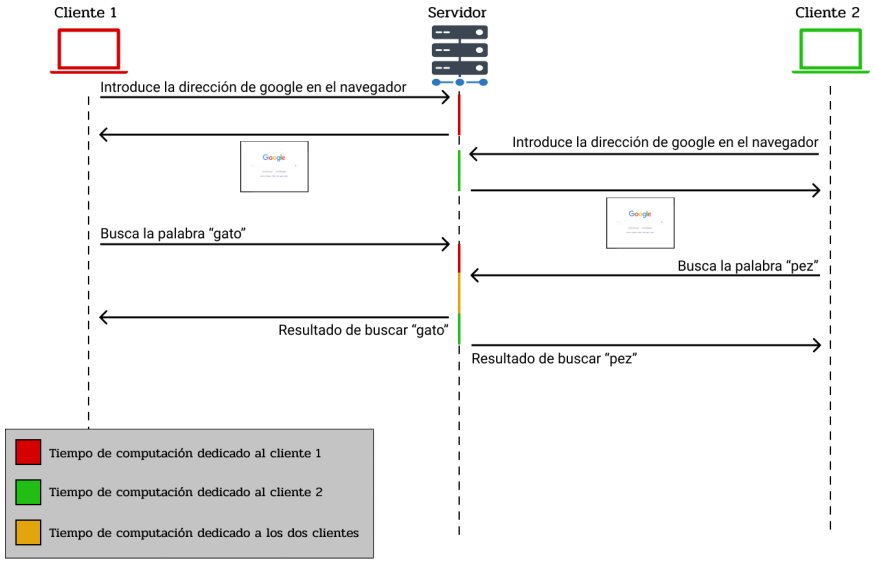

Figura 3.3: Esquema que explica cómo funciona la arquitectura cliente servidor. Fuente: elaboración propia usando [11].

Cabe destacar que los tiempos de ejecución dedicados a cada cliente pasan por varias fases. En primer lugar, se intercepta la solicitud por parte del servidor. En segundo lugar, se procesa la solicitud y se realiza la operación requerida (buscar imágenes, videos, páginas webs, realizar cálculos, etc). Por último, se prepara el resultado y se envía al destinatario correspondiente. Toda la información expuesta durante este apartado ha sido extraída de [11] [12] [13].

3.2.2. Front-end

Front-end hace referencia a la parte del software de una aplicación web que interactúa con el usuario directamente. Este software es ejecutado y/o renderizado por el cliente, dando lugar a interfaces de usuario con las que poder trabajar de forma sencilla. La función principal del front-end es recoger información que el usuario introduce, mandarla al servidor y mostrar los resultados que este devuelve. Los componentes básicos de un front-end son los siguientes:

- **HTML**: del inglés *HyperText Markup Language*, es el elemento más básico dentro de una aplicación web. Define cómo se va a estructurar el contenido de la página. *HyperText* se refiere a los enlaces que conectan unas páginas con otras. *Markup* hace referencia a las etiquetas utilizadas para delimitar textos, imágenes y otros contenidos con el fin de mostrarlos al usuario. Algunas de las etiquetas más comunes son <head>, <title>, <body>, , , <img>, etc. Por último, *Language* quiere decir que es un lenguaje y como tal tiene unas reglas morfosintácticas. Sin embargo, no se debe confundir con un lenguaje de programación, ya que no ejecuta ninguna instrucción en nuestro dispositivo; sólo le dice al navegador cómo mostrar la información.

- **CSS**: es un lenguaje declarativo utilizado para controlar el estilo de la página. Con él conseguimos mejorar de forma significativo el aspecto de la aplicación. Su funcionamiento se basa en definir el aspecto visual que tendrán los elementos contenidos en la página, los cuales son definidos previamente con HTML mediante etiquetas.

- **JavaScript**: es un lenguaje de programación ideado inicialmente para ejecutar pequeños fragmentos de código (“scripts”) en el lado del cliente. Entre sus principales funciones se encuentra hacer la página más interactuable, formatear los datos que se mandarán al servidor y, en ocasiones, comunicarse con el servidor sin necesidad de recargar la página completamente. Esto último recibe el nombre de peticiones AJAX y con ellas conseguimos mejorar la experiencia de usuario evitando tiempos de carga extra.

Una vez entendidos los componentes básicos, vamos a explicar una sería de tecnologías/herramientas que se pueden utilizar en la elaboración del front-end y que nos ofrecen una serie de ventajas.

Angular

Es un *framework* desarrollado en lenguaje TypeScript y creado por Google. Su función principal es proporcionar una serie de herramientas software que facilitan el desarrollo de un front-end utilizando los elementos básicos comentados anteriormente. Utiliza una estructuración estricta aunque eficaz, lo que permite poder escalar proyectos de manera sencilla. Una de las piezas claves de esta estructuración son lo que se denominan componentes. Cada uno define una lógica interna y una plantilla HTML junto con sus correspondientes estilos. La idea con estos componentes es que estén enfocados en cosas específicas y concretas, de manera que puedan ser reutilizados en diferentes partes. Cuando juntamos

todos los componentes y definimos las interacciones entre ellos conseguimos desarrollar un front-end sólido y escalable, otorgando así unas características muy valoradas en el desarrollo de aplicaciones.

Cabe destacar que Angular está desarrollado en TypeScript, y los navegadores actualmente solo pueden interpretar lenguaje JavaScript, por lo que el motor de este *framework*, entre muchas otras cosas, transcribe todo el código a JavaScript para que pueda ser ejecutado por los navegadores. Por otra parte, una de las características interesantes que nos brinda Angular es poder crear aplicaciones SPA que son las siglas de *Single Page Application*. Esto quiere decir que toda la interacción del usuario con la aplicación transcurre dentro de la misma página, sin necesidad de que el navegador esté solicitando cada una de ellas. El funcionamiento es el siguiente: la primera vez que ingresamos a la página nos descarga un único HTML y el JavaScript con las propias instrucciones de Angular. En ese momento ya tenemos todo lo necesario para navegar por la aplicación. Una de las funciones de JavaScript en este caso será ir cambiando dinámicamente el HTML para darnos la sensación de estar navegando, pero ahorrándonos tiempos de carga. Con esto conseguimos tener varias vistas/pantallas sin tener que descargar varias páginas. Sin embargo, no debemos confundir tener una página con tener siempre la misma URL; de hecho, Angular posee un módulo llamado “router” capaz de interpretarla para añadir lógica adicional si fuera necesario.

Por último, al no tener que solicitar más archivos HTML, la comunicación con el servidor se utiliza únicamente para enviar el flujo de datos que se necesiten mostrar de forma dinámica en la aplicación. Esta característica nos permite aliviar el trabajo del servidor, haciendo necesaria solo una interfaz de comunicación entre el front-end y el servidor; esto recibe el nombre de API (*Application Programming Interfaces*).

Generación de gráficos

Mostrar gráficos en una página web es un recurso muy interesante, ya que puede ayudar al usuario a entender mejor la información presentada. En el caso de calculadoras de préstamos es algo muy útil y más aún cuando estos se actualizan de forma dinámica según los datos que introduzca el usuario. Hasta hace unos años la forma más habitual de integrar gráficos en una página era crearlos con una herramienta externa y convertirlos en una imagen estática para mostrarlos en la web. Esto supone un inconveniente; si quieres cambiar el gráfico en función de los datos introducidos por el usuario, tendrías que mandar la información al servidor para que éste la represente y devuelva la imagen al cliente. Esto puede afectar a la experiencia del usuario, por lo que al final se optaba por hacer gráficos estáticos con datos genéricos.

Actualmente, gracias a las mejores computacionales en los dispositivos que contienen los clientes y a las optimizaciones de los lenguajes como JavaScript, se ha trasladado la creación de los gráficos a los clientes. Existen diversas librerías de JavaScript capaces de generar gráficos interactivos de forma rápida a partir de unos datos de entrada. Además, si estos datos cambian las librerías pueden adaptar

los gráficos sin necesidad de recargar toda la página, mejorando así la experiencia de usuario. Algunas de las librerías más utilizadas actualmente son Chart JS, D3 JS y Google Charts.

3.2.3. Back-end

Back-end hace referencia a la parte software de una aplicación web que recibe peticiones de un front-end, las procesa y devuelve una respuesta. En este tipo de aplicaciones, el back-end, se ejecuta en la máquina que contiene el servidor a diferencia del front-end, que se ejecuta/renderiza en la máquina del usuario. Esto puede llevar a preguntarnos lo siguiente; si la máquina del cliente puede ejecutar instrucciones y representar la información, ¿por qué es necesario dividir la lógica de la aplicación en dos o más dispositivos? Es necesario para poder cumplir algunos de los requisitos básicos de una aplicación web, entre los cuales se encuentran los siguientes:

- **Altas prestaciones**: cuando la computación requerida por el servicio requiere de altas prestaciones, es necesario delegar este trabajo al servidor, ya que normalmente estará en una máquina con mejores prestaciones. Con esto conseguimos que las partes más costosas de la aplicación no dependan del rendimiento de los equipos de los usuarios, haciendo posible usar el servicio en dispositivos con bajas prestaciones.

- **Disponibilidad**: el hecho de ejecutar el back-end en diferentes máquinas asegura que si una de ellas falla por algún motivo, tendremos otras disponibles que ofrecerán el mismo servicio, por lo que es muy complicado que la aplicación deje de estar operativa.

- **Almacenamiento de la información**: gracias a que el back-end nos ofrece los datos que necesitamos en cada momento, no hace falta que almacenemos en la máquina del usuario toda la información que maneja la aplicación. Además, si esto pasara y fallara el dispositivo, nos quedaríamos sin la información, perdiéndola en el peor de los casos para siempre. Sin embargo, para que esto no pase los servidores replican la información en diferentes máquinas.

- **Información centralizada**: esto sería imposible si cada cliente almacenara su propia información. Por tanto, gracias a la información centralizada, podemos iniciar sesión en diferentes dispositivos y acceder a la misma información/servicios, independientemente de la máquina que ejecuta el cliente. Una vez descrito qué es el back-end y por qué es necesario, vamos a explicar una sería de tecnologías y módulos que se utilizaran durante el desarrollo del proyecto.

API REST

La evolución de las aplicaciones web durante los últimos años ha contribuido a establecer las API REST como uno de los componentes indispensables. API proviene del inglés *Application Programming Interfaces* que hace referencia a un conjunto protocolos utilizados para comunicar diferentes procesos software. Estos procesos pueden estar o no ejecutándose en la misma máquina.

API REST (*Representational State Transferes*) un tipo de API que define la comunicación entre un cliente y un servidor a través del protocolo HTTP. En la mayoría de los casos la información viaja en el cuerpo de las peticiones, siguiendo algún estándar, como por ejemplo JSON, XML, etc.

Como muy bien se nos indica en [14], para que realmente una API sea considera de tipo REST tiene que cumplir los siguientes requisitos:

- Seguir una arquitectura cliente-servidor utilizando el protocolo HTTP.
- Mantener una comunicación sin estado, es decir, el servidor no almacenara información referente al cliente. Por lo tanto, todas las peticiones son independientes unas de otras.
- Guardar en caché algunas de las interacciones más repetidas con el fin de optimizar el servicio.
- Mantener la misma interfaz de comunicación para todas las solicitudes posibles, de manera que la información se transmita de forma estandarizada.
- Establecer una jerarquía en el servidor, donde se diferencien elementos como los balanceadores de carga, los encargados de la seguridad, etc.

Librería de cálculo

Hasta ahora hemos visto por qué es necesario tener un back-end para utilizar nuestras aplicaciones y cómo nos comunicamos con él. El siguiente paso es procesar las peticiones del cliente y devolverle un resultado. En el caso de las herramientas de cálculo de créditos es necesario codificar todos los procedimientos matemáticos, los cuales serán posteriormente utilizados para calcular aquello que nos solicite el cliente. Todos esos procedimientos son almacenados en conjunto bajo el nombre de librería. Las librerías son un recurso muy común en la programación y nos sirven para ofrecer diferentes niveles de abstracción. En este caso, la librería de la que estamos hablando realiza cálculos matemáticos, pero existen variedad de ellas con funciones muy diversas que pueden ir desde leer de teclado hasta generar gráficos interactivos.

Por otra parte, el número de las librerías que están disponibles en internet con relación a cálculos financieros/matemáticos es muy reducida, por lo que en la mayoría de los casos tiene que ser codificada desde cero, como ocurrirá en el desarrollo de este proyecto.

.NET

Una vez tenemos claro cómo se va a comunicar el cliente con el servidor y que operaciones va a ejecutar el servidor, necesitamos hacerle llegar las peticiones de forma ordenada y que éstas desencadenen únicamente la lógica interna solicitada por el cliente. Para ello vamos a usar un Framework, más concretamente ASP.NET, que pertenece a la implementación de .NET Framework. Esta implementación, entre muchas ventajas, nos ofrece una gran compatibilidad con los servicios de Azure, facilitando

así el despliegue de aplicaciones.

Dentro del entorno .NET tenemos dos grandes implementaciones: por una lado encontramos .NET Framework y por el otro .NET Core. Algunas de sus características son:

- .NET Framework: creado por Microsoft en 2002, es una de las más extendidos en la actualidad y posee una gran cantidad de documentación en internet. Sin embargo, en sus inicios fue creada únicamente para ser ejecutado en sistemas operativos de Microsoft; actualmente esto no ha cambiado y solo es posible usarlo con sistemas operativos Windows. Dentro del ámbito de servicios web, cuenta con ASP.NET, uno de los entornos para aplicaciones web más extendidos.

- .NET Core: esta implementación fue lanzada en 2016 por Microsoft. Su intención con ella fue poder brindar a los desarrolladores la posibilidad de ejecutar las aplicaciones en cualquier sistema operativo. Por su parte, también cuenta con su entorno ASP.NET Core para desarrollar servicios web.

A pesar de las diferencias entre ambas implementaciones, también tienen cosas en común, ya que ambas pertenecen al entorno .NET. Por ejemplo, en ambas implementaciones es posible escribir código en los lenguajes C#, VB y F#. Esto es posible porque .NET compila en primera instancia el código de alto nivel traduciéndolo a un lenguaje intermedio. Posteriormente este código en lenguaje intermedio será utilizado por la máquina virtual de .NET que se encargará junto con un JIT-Compiler (*"Just in time compiler"*) de crear el código máquina que se ejecutará en el equipo. A pesar de que la máquina virtual no es exactamente la misma para ambas implementaciones, sí que utilizan el mismo compilador (Roslyn) [15]. Además, al utilizar un lenguaje intermedio creado a partir del mismo compilador, muchas de las librerías que utiliza .NET Framework también las utiliza .NET Core.

La información utilizada para explicar este apartado ha sido obtenida a partir de [16] [17].

3.2.4. Patrón MVC

Normalmente se suele hacer la separación entre front-end y back-end a la hora de hablar de una aplicación web. Esto es debido a que son conceptos fáciles de entender y ejemplifican muy bien el funcionamiento lógico. Sin embargo, si profundizamos nos damos cuenta que desde el punto de vista "lógico" existen más separaciones posibles. Uno de los patrones de diseño más utilizado para los servicios web es el denominado patrón MVC (*Model-View-Controller*). En él se distinguen 3 componentes principales: modelo, vista y controlador. Utilizando este diseño se puede conseguir que cada componente se encargue de cosas diferentes; esto es muy importante, porque cuando crece la complejidad de la aplicación y no se sigue un buen diseño, el desarrollo se vuelve insostenible. El patrón funciona de la siguiente manera: cuando el usuario solicita un recurso (imagen, cálculo, acción) interactúa en primer lugar con el controlador, que se encarga de comunicar al modelo qué acción ha de realizar. Una vez el modelo la ha realizado, devuelve el resultado al controlador y este decidirá qué vista crear para mostrársela al usuario. Según [18] las funciones de los componentes en un servicio web son:

- **Modelo**: su objetivo es representar el estado de la aplicación y llevar a cabo la lógica de negocio. Esta lógica ha de ser encapsulada en el modelo junto con cualquier herramienta que permita la persistencia de datos en caso de ser necesario.
- **Vista**: su finalidad es representar el resultado en la interfaz de usuario. La lógica presente en la generación de las vistas ha de mínima y siempre tiene que estar relacionada con la representación del contenido.
- **Controlador**: su función es manejar las interacciones/solicitudes del usuario, comunicarse con el modelo y seleccionar que vista renderizar para mostrar los resultados.

En la actualidad hay muchos frameworks que facilitan el uso de estos patrones como Django (Python), Flask (Python), Lavarel (PHP), etc. En este caso y aprovechando la elección de la implementación .NET Framework se ha utilizado el framework ASP.NET. Al añadir al proyecto el framework de Angular, el patrón queda un poco más difuso, porque al usar el modelo de SPA estamos usando dos controladores a la vez, uno en Angular y otro en ASP.NET. Del mismo modo, tendríamos dos vistas, la que devuelve el controlador de ASP.NET (contiene Angular) y la propia vista que genera Angular, que será la que acabe mostrando el navegador.

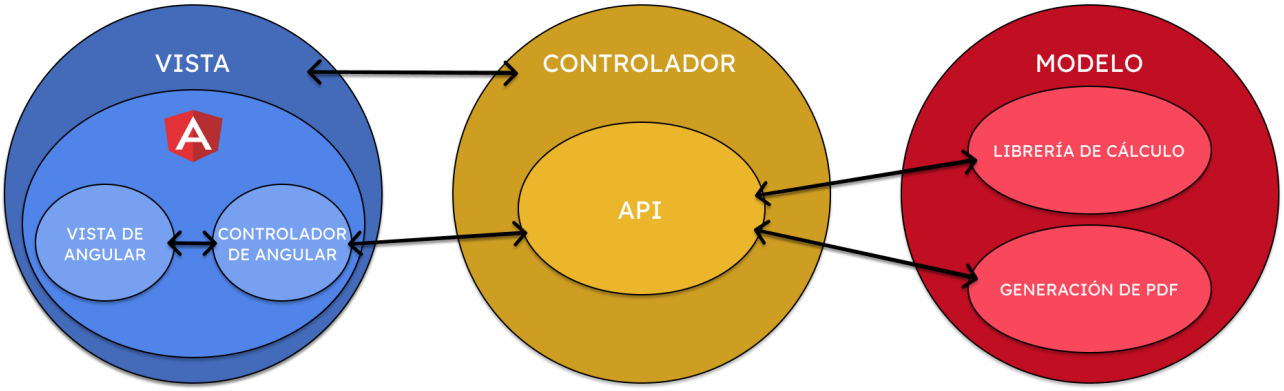

Figura 3.4: Implementación del patrón MVC junto con Angular. Fuente: elaboración propia.

3.3. Mi propuesta

Como se ha explicado durante el capítulo 2, el cálculo de los flujos de amortización de un crédito no es algo que se pueda hacer de forma sencilla, por lo que resulta buena idea automatizar este proceso. Cualquier persona con unos conocimientos en economía podría calcularlo utilizando Microsoft Excel, el problema surge cuando la mayoría de personas que solicitan estos créditos no tiene los suficientes conocimientos de economía. En ese caso es útil crear una herramienta que automatice el proceso y

devuelva los resultados de forma que cualquier persona pueda entenderlos.

Debido a las diferentes ventajas que tiene realizar una aplicación web, mi propuesta opta por utilizar este modelo de aplicación, usando las tecnologías expuestas anteriormente. Previamente se ha comentado que la cantidad de librerías de cálculo para estos casos tan concretos es muy poca o nula. Por lo tanto, se optará por crear una librería utilizando conceptos de programación orientada a objetos y algoritmos computacionales, los cuales hagan un uso eficiente de los recursos. También se hará uso de los patrones y arquitecturas mencionadas con anterioridad, con el fin de obtener una aplicación correctamente estructurada. El hecho de utilizar un buen diseño hace más sencillo que la aplicación sea escalable y que pueda crecer tanto en funcionalidad como en el número de servicios que pueda proporcionar de forma paralela. Desde el punto de vista de la experiencia de usuario, vamos a implementar lo que se conoce como una SPA (*Single-page application*), lo que permite que el usuario pueda navegar por la página sin necesidad de recargar la misma. Además, las SPA realizan una gestión del tráfico de red más eficiente, ya que solo se comunican con una API para solicitar exclusivamente datos y no otro tipo de documentos más pesados, como pueden ser páginas HTML. Por último, respecto al despliegue del servicio en internet se ha optado por utilizar los servicios de Azure, que junto con el proceso de integración continua facilitan la detección temprana de fallos.

Por otra parte, el desarrollo de la aplicación se va a enfocar en cumplir el marco legal vigente y todas las premisas que se aprobaran a lo largo del año 2021. Entre ellas se encuentran generar información precontractual con ejemplos representativos, la diferenciación clara de los intereses y de la amortización total, informar de posibles endeudamientos perpetuos, etc. Asimismo, y con el fin de adaptarse a las diferentes modalidades de pago que ofrecen actualmente las entidades financieras, se implementarán 3 variantes del cálculo, de forma que el usuario pueda ver cuál de ellas se adapta mejor a sus necesidades. Otra de las características que diferenciará este proyecto de los actualmente existentes, será la implementación de una interfaz amigable y la posibilidad de interactuar con gráficos.

4

Diseño, implementación y pruebas

Anteriormente se ha explicado qué funcionalidad tiene que reunir la aplicación web y qué tecnologías se van a usar para ello, por tanto, lo último que nos queda por realizar es la implementación. A lo largo de este capítulo vamos a ver los detalles de ésta y las pruebas realizadas para comprobar el correcto funcionamiento del servicio. También se explicará el proceso de despliegue que se ha llevado a cabo, el cual se inicia cuando el código de la aplicación está en el repositorio de versiones y termina cuando el servicio es accesible desde internet. Además, explicaremos cómo hemos otorgado seguridad a la API REST para que no sea accesible desde fuera de la aplicación.

4.1. Diseño arquitectónico de la aplicación

Para explicar de forma sencilla la arquitectura que utiliza la aplicación y la función qué desempeña cada uno de sus módulos, hemos desarrollado el siguiente esquema.

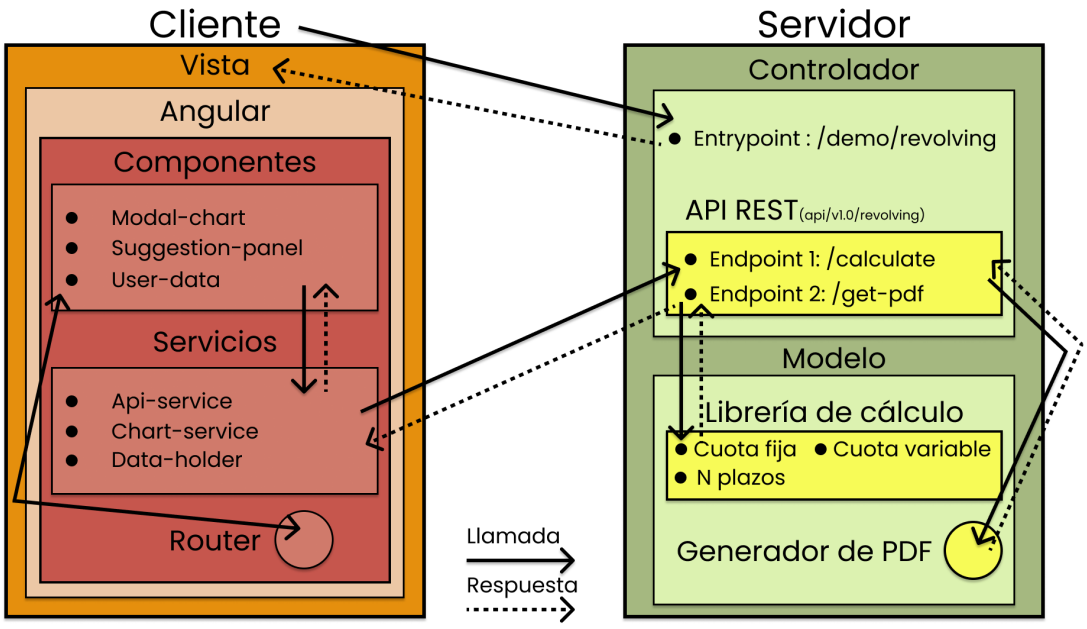

Figura 4.1: Arquitectura de la aplicación web. Fuente: elaboración propia.

En primer lugar, el cliente le solicita al controlador mediante un "*Entrypoint*" (identificador del recurso solicitado) la vista principal de la página. En ese momento, como hemos explicado anteriormente, el servidor devuelve embebido en la vista todo el código JavaScript relacionado con Angular. Dentro este framework, podemos diferenciar tres grandes grupos de módulos.

4.1.1. Componentes

Antes de ser compilados a código JavaScript, los componentes son un conjunto 3 ficheros. Un fichero HTML donde se definirá la estructura del DOM de ese componente, un fichero CSS que definirá el aspecto que tendrá ese HTML y por último un fichero TypeScript que define la lógica interna del componente. En otras palabras, son pequeños fragmentos de la página que tienen lógica propia y pueden ser reutilizados cuantas veces se quiera. En este proyecto los 3 componentes principales son los siguientes:

- **User-data**: se encarga de procesar los datos introducidos por el usuario e invocar al servicio "Api-service" para que realice la petición. Una vez se han devuelto los resultados, este componente se encarga de mostrarlos al usuario. También se encarga de actualizar el contenido del servicio "Data-holder".

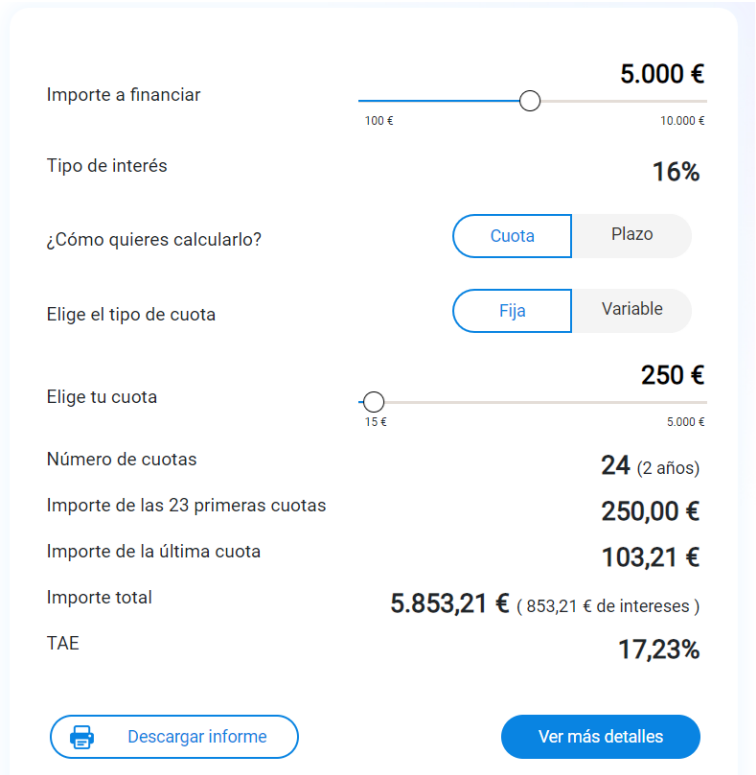

Figura 4.2: Componente user-data. Fuente: elaboración propia.

- **Suggestion-panel**: como comentamos anteriormente, cuando la cuota elegida no supera el 25 % del total solicitado es necesario mostrarle al usuario otras alternativas de devolución. Ésta es precisamente la función de este componente y su funcionamiento se basa en observar el servicio “Data-holder”, cuando éste contiene un cálculo en el cual la cuota elegida no satisface ese 25 %, “Suggestion-panel” solicita mediante el “Api-service” cuatro nuevos cálculos con las cuotas correspondientes que marcaría la legislación. En caso contrario, cuando se supera el 25 % este componente dejará de ser visible para el usuario y no solicitará ningún cálculo.

Figura 4.3: Componente suggestion-panel. Fuente: elaboración propia.

- **Modal-chart**: su función es abrir un modal (una pequeña ventana) que se superponga al contenido de la página. En él se muestran dos gráficos, el primero de ellos es una composición de barras apiladas alineadas horizontalmente, en el que se representa la diferencia entre el capital a amortizar y los intereses totales que se pagarán. El segundo gráfico, se encarga de mostrar cómo evolucionará la cuota a lo largo del préstamo, separando las partes correspondientes a los intereses y la amortización. Cada uno de los gráficos está en su propio componente y toman los datos del servicio “Data-holder” para representar la información.

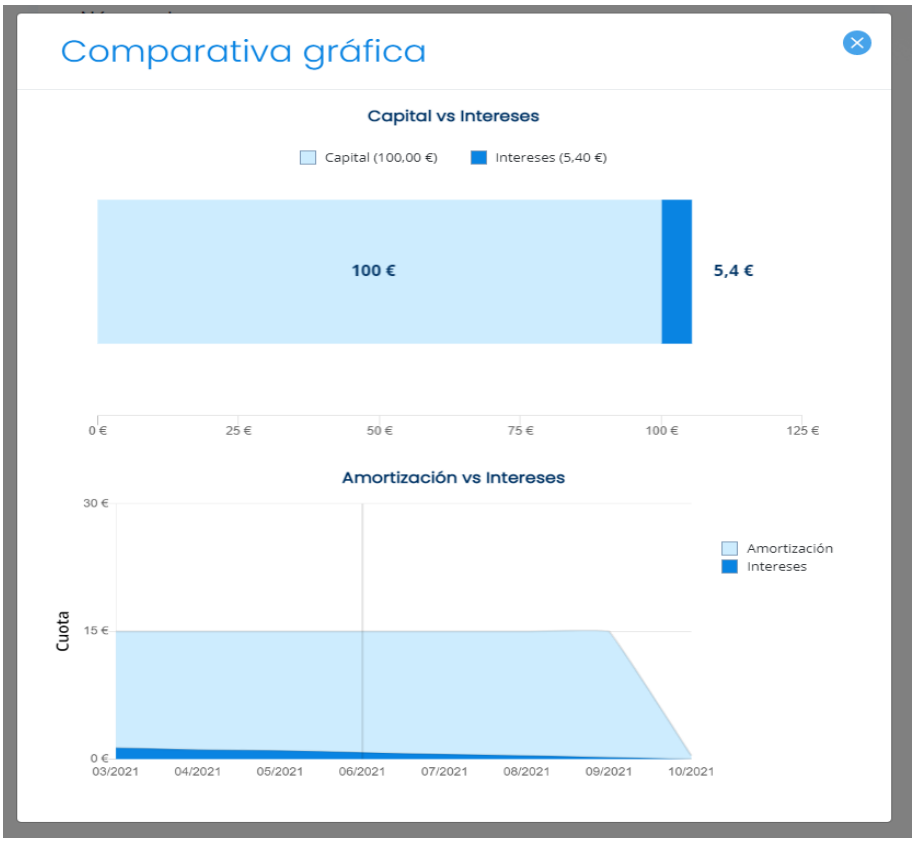

Figura 4.4: Componente modal-chart. Fuente: elaboración propia.

4.1.2. Servicios

A diferencia de los componentes, los servicios no tienen que manejar nada relacionado con el apartado visual. Una de sus funciones es contener la lógica que tienen en común los diferentes componentes, de forma que no se repitan los mismos procedimientos en cada uno de ellos.

- **Api-service**: como su propio nombre indica, este servicio se encarga de comunicarse con la API mandando los datos en formato JSON. Su estructura se basa en dos funciones principalmente; cada una de ellas recibe los "*inputs*" correspondientes y solicita a la API el recurso requerido. Por último, el resultado es devuelto al componente que haya invocado a estas funciones. Cabe destacar que dentro de la funcionalidad de este servicio, se contempla la posibilidad de que la comunicación falle; en ese caso se redirigirá al usuario a la pantalla correspondiente.
- **Data-holder**: la función de este servicio es mantener ciertos datos accesibles desde cualquier componente. Esto permite que puedan suscribirse a datos externos y que sean notificados cuando cambien; una vez llegue la notificación cada componente desencadenará la lógica correspondiente. Por ejemplo, User-data llama al Api-service y guarda el resultado (del cálculo) en el Data-holder; esto permite que el componente Suggestion-panel pueda suscribirse a ese resultado, de manera que cuando este dato cambia, Suggestion-panel es

notificado y decide si realizar sus propios cálculos.

- **Chart-service**: la configuración de los diferentes gráficos la realiza este servicio. Para ello, define para cada gráfico características propias como ancho, alto, tipo de gráfico (barras, áreas, etc.), colores, configuración de los ejes, etc. Este servicio es usado en dos ocasiones principalmente, la primera para mostrar los gráficos en la propia página, la segunda para mandar la configuración al servidor y que este se encargue de mostrar los gráficos correspondientes en el informe.

4.1.3. Router

El router es un módulo propio de Angular totalmente configurable. Su función es interpretar la URL y mostrar los componentes correspondientes en cada caso. Al igual que ocurre con los servicios, el router también es accesible por los componentes, por lo que es útil usarlo para redirigir a otras vistas/componentes o leer la propia URL desde el componente. Una utilidad de poder leer la URL es poder parametrizar argumentos, por ejemplo, para determinar el idioma de la página.

Una vez el cliente tiene la SPA ejecutándose, solo se comunicará con la API para solicitar información o recursos. Cabe destacar que solo son necesarios dos "*endpoints*" para usar toda la funcionalidad de la aplicación. Cada uno de ellos se encargará de comunicarse con la parte del modelo correspondiente. No obstante, en este apartado no vamos a profundizar en el funcionamiento del modelo, ya que a lo largo de este capítulo vamos a ver más en detalle cómo se ha implementado y las decisiones de diseño.

4.2. Front-end adaptable a diferentes dispositivos

Anteriormente hemos comentado las ventajas de desarrollar una aplicación web frente a otros tipos de aplicaciones convencionales. Entre las ventajas se encontraba que, con una única aplicación, puedes ofrecer servicios a todos los usuarios que dispongan de un navegador (ordenador, tableta, móvil, etc.). El problema surge al desarrollar una interfaz que se adapte a diferentes tamaños de pantalla. Por ello, vamos a explicar cómo se ha llevado a cabo el proceso de adaptación del contenido en función del tamaño de pantalla.

Las *"Media queries"* son una de las directivas básica que proporciona CSS para poder construir interfaces adaptables. Éstas han sido utilizadas en diferentes elementos de la página, por ejemplo, en los tamaños y posiciones de las fuentes, iconos, etc. En el apéndice A.1.1 podemos ver un ejemplo de cómo se han utilizado.

Para el reposicionamiento de elementos más complejos, como pueden ser las sugerencias o los conjuntos de etiqueta y campo a rellenar por el usuario, se ha utilizado el framework de Bootstrap.

Su uso es bastante sencillo, pues basta con añadir clases (predefinidas por Bootstrap) a los elementos del archivo HTML y automáticamente se aplicarán las propiedades correspondientes. Sin embargo, una de las características más potentes de este framework es su sistema *"grid"* por filas y columnas, de forma que, según el tamaño de la pantalla, se pueda establecer que haya x columnas o x-1 y que la columna que falta pase a la fila de abajo. A continuación podemos ver un ejemplo para la parte de las sugerencias cuando la pantalla es más pequeña que la de un monitor habitual.

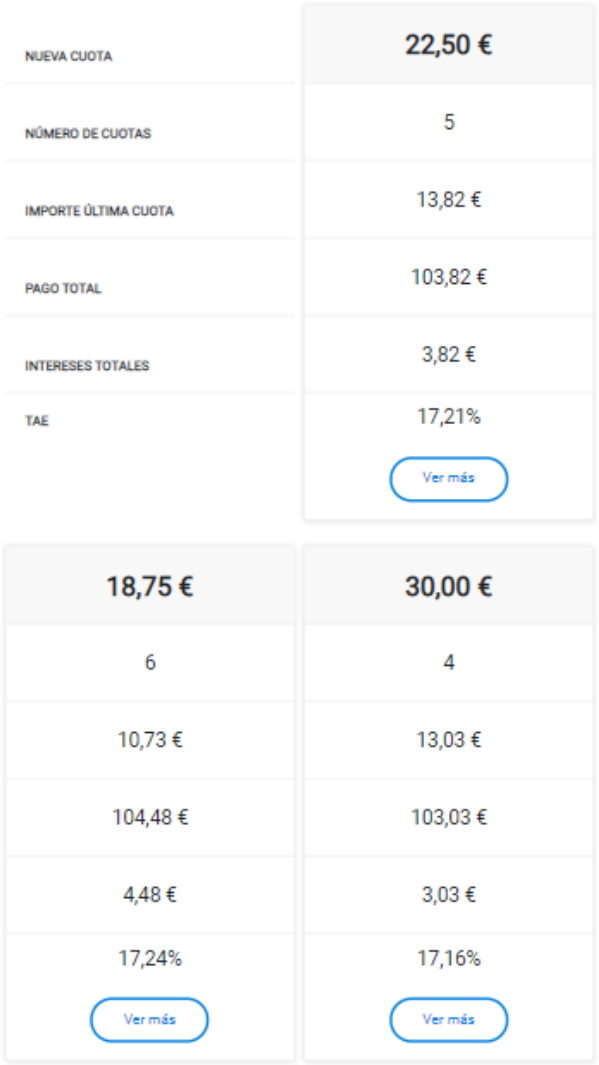

Figura 4.5: Panel de sugerencias en dispositivos pequeños. Fuente: elaboración propia.

En el apéndice A.1.2 podemos ver un ejemplo de Bootstrap que ilustra este comportamiento.

4.3. Protección de la API REST mediante un token JWT

Para que el servicio de cálculo y la generación de PDF no sean accesibles desde el exterior, se ha optado por hacer uso de tokens JWT (*"JSON Web Tokens"*). A pesar de que ya existen librerías

que realizan el proceso de creación y comprobación de estos tokens para diferentes lenguajes, se ha decidido implementar desde cero para comprender mejor el funcionamiento de este. Cabe destacar que la función criptográfica HMAC utilizada en el proceso sí pertenece a la librería de .NET Framework, por lo que no ha tenido que ser implementada.

A continuación, explicaremos cómo se realiza el proceso de envío y renovación del token. En primer lugar, cuando el cliente realiza la primera solicitud, el controlador envía tanto la primera vista como un token con una fecha de expiración de 15 minutos. Posteriormente cada vez que el cliente quiere realizar un cálculo, envía el token en la cabecera de la petición para que el controlador pueda verificarlo. Todas las peticiones relacionadas con los cálculos que llegan a la API pasan primero por una función que verifica la validez del token (haciendo uso de nuestra implementación). Si el token es válido la petición continua en el *endpoint* correspondiente, en caso contrario se devuelve un error 401 (*Unauthorized*). Cuando se devuelve este error, el servicio Api-service descrito anteriormente se encarga de redirigir al usuario a la pantalla de error correspondiente. Sin embargo, cuando el token es válido continua con normalidad la ejecución del servicio. Además, antes de devolver el resultado al cliente, se genera otro token con 15 minutos más. De esta forma conseguimos una auto-renovación del mismo de forma sencilla.

En el caso de la generación de informes no funciona exactamente igual. Esto se debe a que usamos un pequeño módulo de Angular que se encarga de mandar la petición en otra pestaña diferente del navegador. Además, dicho módulo no permite introducir el token en la cabecera; por ello lo mandamos en el cuerpo de la petición. Cuando ésta se envíe pasará directamente al *endpoint* correspondiente, que tendrá que ocuparse de validar el token y actuar en consecuencia (generando un informe o devolviendo un error).

Para entender mejor cómo se ha implementado el manejo del token (creación y validación), podemos ver en el apéndice A.2 el pseudocódigo que ilustra el funcionamiento real que sigue la aplicación.

4.4. Librería de cálculo

Esta librería se ha desarrollado con el fin de hacer el cálculo lo más cómodo posible para el controlador, de forma que solo tenga que introducir unos parámetros de configuración del cálculo (enviado previamente por el cliente) y ejecutar una única función. El resultado de los cálculos será enviado al cliente sin ningún tipo de procesamiento intermedio.

Cabe destacar que, durante el desarrollo de esta librería se han tenido en cuenta las posibles convenciones de pagos para días festivos, que pueden ser: PBD (*Previuous Business Day*) y FBD (*Following Business Day*). Sin embargo, estas convenciones han sido implementadas de forma que solo tengan en cuenta los días pertenecientes al fin de semana. Por otra parte, se permiten dos posibles bases de cálculo: 30/360 y actual/360 (explicadas en el capítulo 2).

En el apéndice A.3 podemos ver el diagrama de clases UML que se ha seguido durante la implementación de la librería.

Si nos fijamos en las clases que heredan de “RevolvingCalculator”, hay una de ellas (“FixedTermCalculator”) que no calcula intereses, cuotas ni amortización. De hecho, si recordamos cómo se realizaba el cálculo con N plazos no es necesario que lo haga. Para realizar esta modalidad, se utiliza el cálculo de cuota fija y se varía la cuota mensual hasta obtener un resultado con los meses deseados. En el mundo de la computación hay muchas formas de “ir probando valores” hasta que se encuentra el resultado deseado; de hecho, es un problema bastante recurrente. En este caso, se ha decidido atacar utilizando el algoritmo de búsqueda binaria, ya que no es complejo de implementar y es uno de los más rápidos cuando los datos están ordenados. Cabe destacar que, en este caso, no estamos buscando valores en una tabla propiamente dicha, pero la función de cálculo con cuota fija se comporta de forma muy similar, ya que a medida que aumentamos la cuota disminuye el número de meses y viceversa, es decir, es una función decreciente simple. Para este caso, el valor máximo de la cuota será equivalente a la cantidad total solicitada y el valor mínimo a 1 €. En el apéndice A.4 podemos ver el pseudocódigo que ilustra cómo se ha implementado.

También se ha hecho uso de este algoritmo para hallar el valor de la TAE; más concretamente se ha utilizado para modificar el valor de la TAE hasta conseguir que el VAN sea igual a 0, cómo se explicó en capítulo 2.

4.5. Generación de PDF

Una vez se ha comprobado la validez del token comienza el proceso para generar el informe. En primer lugar, se realizan de nuevo los cálculos con los datos de entrada enviados por el cliente; esto lo hacemos porque si usáramos los resultados que nos envía el cliente, podríamos estar mostrando informes erróneos/manipulados.

En segundo lugar, necesitamos transformar la configuración de los gráficos (que se envía en la petición) a imágenes que se puedan plasmar en el informe; para ello utilizamos un pequeño servicio web desarrollado por la entidad que colabora en el desarrollo de este proyecto. Este servicio ejecuta un pequeño servidor de NodeJs capaz de interpretar la configuración del gráfico (previamente enviada) y devolver una imagen de éste. Si bien es cierto que se podría hacer una captura a los gráficos que posee el cliente y enviarlos directamente, esto no es del todo seguro ya que, como hemos comentado anteriormente, el usuario podría modificar los datos y hacernos generar un informe incorrecto.

Por último, restaría juntarlo todo (el resultado de los cálculos, los gráficos y algunas imágenes o textos). La librería que vamos a utilizar para llevarlo a cabo es AspPdf, que mediante directrices escritas en C# nos permite distribuir los elementos y definir las características de los mismos. También nos permite establecer algunas propiedades del archivo PDF como el título y el autor. Una vez es

procesado el informe, la propia librería nos devuelve un “Array” de Bytes que será devuelto al cliente.

Cabe destacar que, para facilitar el uso al controlador se ha desarrollado una clase que contiene toda la funcionalidad relacionada con el último paso. Además, dentro de la misma clase se declaran las rutas de los directorios donde se encuentran las fuentes e imágenes que serán utilizadas, todo ello con el fin de hacerlo lo más modular posible.

En el apéndice A.5 podemos ver un informe generado con la herramienta web.

4.6. Pruebas realizadas

Las pruebas son un elemento indispensable a la hora de desarrollar software de calidad. Por ello, en este proyecto se ha dedicado una parte importante del tiempo a probar el correcto funcionamiento de la aplicación. Las pruebas llevadas a cabo durante y después del desarrollo se pueden agrupar en los siguientes tipos:

- **Pruebas unitarias**: éstas se han centrado en comprobar el correcto funcionamiento de la librería de cálculos; para ello se ha creado dentro de la solución de Visual Studio un proyecto de test unitarios. De esta forma, podemos ejecutar todos los test de forma automática una vez estén creados. Para la validación de los resultados, se ha utilizado un Excel creado por el Área de Banca de la entidad que colabora en este proyecto. También se han simulado casos en los que el usuario introduce parámetros erróneos; de esta manera comprobamos que la aplicación no va a dejar de funcionar en caso de recibir datos no contemplados. Cabe destacar que todas estas pruebas serán ejecutadas cada vez que se suban cambios a la herramienta de Git, gracias al proceso de integración y distribución continua.
- **Pruebas funcionales**: con el fin de comprobar el correcto funcionamiento de la interfaz de usuario, cada vez que se añadía/modificaba alguna característica se probaba de forma manual todas las posibles acciones que podría llevar a cabo el usuario. De esta forma, localizábamos posibles fallos de interacción entre los diferentes componentes.
- **Pruebas de seguridad**: éstas se han basado en comprobar el correcto funcionamiento del token JWT. Para llevarlas a cabo se ha utilizado una aplicación externa denominada “Postman”. Con ella se mandaban diferentes tokens (dentro de las peticiones) a la API REST, algunos de ellos completamente válidos y otros modificados con el fin de simular tokens creados de forma externa.

Gracias a la realización de estas pruebas durante y después del desarrollo hemos sido capaces de detectar errores de forma temprana. Esto es algo muy positivo ya que nos ayuda a no arrastrar errores en el código, los cuales pueden ser cada vez más difíciles de solucionar a medida que aumenta la complejidad de la aplicación.

4.7. Despliegue de la aplicación

Una vez tenemos toda la aplicación desarrollada y sabemos que funciona correctamente en el entorno de desarrollo, es el momento de desplegarla para que sea accesible a través de internet. Primeramente, vamos a explicar la configuración previa necesaria y posteriormente detallaremos el propio proceso de despliegue, que abarca desde que se sube el proyecto a la herramienta Git hasta que el servicio termina siendo accesible desde un navegador. Cabe destacar que, para realizar todo este proceso de despliegue, se ha elegido la plataforma Cloud de Azure.

En primer lugar, debemos dar de alta el servicio "App-Service" en la propia plataforma Cloud; sin él no tendríamos ningún sitio en el que ejecutar nuestra aplicación. Una vez tenemos el servicio contratado Azure nos proporciona una URL automáticamente. Sin embargo, esta dirección es un subdominio de Azure, por lo que le vamos a solicitar al departamento de sistemas que dé de alta otro dominio que utilice un registro de tipo CNAME. El hecho de tener una URL que redirija a la de Azure nos proporciona un DRP (*Disaster Plan Recovery*), ya que en caso de que los servidores de Europa no estuvieran operativos, bastaría con cambiar la región de despliegue del servicio. Después, tenemos que configurar los pipelines en la herramienta DevOps; en ellos se especifica una serie de pasos entre los que se encuentran descargar el proyecto del repositorio, descargar las librerías necesarias, compilar el proyecto, pasar los test previamente creados e incluidos en la solución, comprimir el resultado, etc. Este pipeline, será ejecutado cada vez que se realice un push a la rama de Git seleccionada, gracias al proceso de integración continua. Para finalizar la configuración, tenemos que crear otro pipeline más específico llamado *release*, en el que se realizarán cambios en las variables de entorno del programa, por ejemplo, direcciones a las que conectarse, ya que al salir del entorno de desarrollo no tenemos que conectarnos más a localhost. Por último, en el release, se crea una instrucción específica que se encargará de mandar todo el código compilado y funcional al "App-Service" de Azure para que se pueda ejecutar allí.

Una vez tenemos todo configurado podemos explicar el proceso de integración y distribución continua que se lleva a cabo. Todo el proceso comienza cuando se realiza algún cambio en la rama configurada previamente. Tras confirmarse los cambios, la propia aplicación de DevOps los detecta y crea una tarea (ésta contiene en formato YAML todas las instrucciones del pipeline inicial configurado previamente). Dicha tarea se introduce en una cola, donde esperará hasta ser recogida por un *build agent* (software que corre en una máquina ajena a DevOps). Después, este agente se encarga de ejecutar las indicaciones descritas en el fichero YAML. Cuando acaba de ejecutarse el primer pipeline, DevOps lo detecta y encola otra tarea con las instrucciones del release. Dicha tarea será ejecutada por el mismo build agent que realizó el pipeline anterior. Finalmente, la aplicación quedará desplegada tras ejecutar la última instrucción del release, por lo que ya sería accesible desde internet.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

El objetivo de este proyecto ha sido crear una herramienta sencilla e intuitiva para calcular créditos revolving. Así, los usuarios que quieran contratar este tipo de créditos, tendrán la posibilidad de usar esta aplicación y ver de forma clara cómo se va a desarrollar la amortización a lo largo del tiempo. Además, esta herramienta cumple con los requisitos legales vigentes y con los que entrarán en vigor a lo largo del año 2021.

Debido a que queríamos enfocar la herramienta a todos los públicos, la primera decisión de diseño que tomamos fue optar por una aplicación web; de esta manera, eliminábamos cualquier dificultad que pudieran tener los usuarios a la hora de descargar/instalar la aplicación en los dispositivos. Posteriormente y con el fin de crear una buena experiencia de usuario junto con la menor interacción posible con el servidor, decidimos utilizar el modelo de SPA para desarrollar el front-end. Debido a esto, tuvimos que crear una API REST que recibiera las peticiones del front-end. Además, tanto para el desarrollo de la API como del resto de partes del back-end se ha utilizado el framework de ASP.NET de la implementación de .NET Framework, lo que ha contribuido a seguir un esquema organizado y crear una aplicación escalable. Asimismo, aprovechando la posibilidad ofrecida por la entidad que colabora en este proyecto, se ha desplegado la aplicación web haciendo uso de los servicios Cloud de Microsoft. De esta forma, evitamos los problemas de disponibilidad debidos al hardware y estar configurando las máquinas para el despliegue, ya que usamos servicios del tipo PAAS.

Por otra parte, creemos que hemos realizado un buen trabajo al cumplir los requisitos planteados, tomar buenas decisiones de diseño, utilizar tecnologías Cloud y dedicar una parte del desarrollo a la realización de pruebas. Con todo esto, hemos creado una aplicación robusta frente a fallos y a la vez escalable, que permite introducir cambios sin demasiado esfuerzo. Cabe destacar que, no se había tenido contacto previo con ninguna de las tecnologías utilizadas, por lo que se ha realizado un esfuerzo importante durante el desarrollo. Dentro de los conocimientos adquiridos podemos destacar el uso de ASP.NET, C#, .NET Framework, TypeScript, Angular, ChartJs, Azure, AspPDF, Devops, Bootstrap y la configuración del modelo de integración y distribución continua.

5.2. Futuras ampliaciones y mejoras

Si bien creemos que hemos realizado un buen trabajo, hay ampliaciones que podrían realizarse con el fin de añadir nueva funcionalidad o mejorar el rendimiento de la aplicación. Por ello, vamos a listar algunas mejoras o ampliaciones que podrían realizarse en un futuro:

- **Mejorar la gestión del token**: actualmente el token sólo se renueva cuando el usuario solicita un nuevo cálculo. Sin embargo, sería interesante en un futuro gestionar en el front-end el tiempo restante del token para solicitar otro nuevo cuando éste está a punto de caducar.
- **Nuevas bases de cálculo**: la librería de cálculo desarrollada solo contempla dos bases 30/360 y Actual/360. Además, la aplicación está configurada para usar únicamente una de ellas; por tanto, sería muy útil ampliar este apartado en el futuro, con el fin de añadir todas las bases de cálculo a la librería y dejar al usuario elegir cuál de ellas usar.
- **Mejorar el rendimiento de la librería de cálculo**: la librería utilizada en la aplicación está desarrollada para ejecutarse en un único hilo. Sin embargo, podría ser interesante estudiar las dependencias internas de los cálculos para realizar una implementación que soporte ejecutarse en diferentes hilos, de forma que se mejore el rendimiento general.

Bibliografía

[1] "Crédito revolving," *elEconomista*, 2015. https://www.eleconomista.es/diccionario-de-economia/credito-revolving. Consultado: 10/04/2021.

[2] "¿Qué son las tarjetas revolving?," *ABC*, p. 1, 2020. https://www.abc.es/economia/abci-tarjetas-revolving-202003041316_noticia.html. Consultado: 10/04/2021.

[3] Tribunal Supremo, "Sts 600/2020," pp. 1–8, 2020.

[4] Ministerio de Asuntos Económicos y Transformación Digital, "Orden etd/699/2020, de 24 de julio," no. 203, pp. 58048–58063, 2020.

[5] Ministerio de Gracia y Justicia, "Ley de 23 de julio de 1908 sobre nulidad de los contratos de préstamos usurarios," no. 206, pp. 1–4, 1908.

[6] R. V. Burguillo. "Flujo de caja financiero (FCF)," https://economipedia.com/definiciones/flujo-de-caja-financiero.html. Consultado: 10/04/2021.

[7] Bankinter. "¿Qué es la TAE y para qué sirve?" https://www.bankinter.com/banca/preguntas-frecuentes/hipotecas/que-es-la-tae-y-para-que-sirve. Consultado: 10/04/2021.

[8] EVO Banco. "¿Que es la TAE de un préstamo?" https://www.evobanco.com/ayuda/al-dia-con-EVO/prestamos/tae-que-es/. Consultado: 10/04/2021.

[9] D. S. D. W. K. J. Raj Bala, Bob Gill, "Magic quadrant for cloud infrastructure and platform services," *Gartner*, 2020.

[10] "Soluciones de DevOps en Azure," https://azure.microsoft.com/es-es/solutions/devops/. Consultado: 10/04/2021.

[11] G. Reese, *Database Programming with JDBC Java: Chapter 7. Distributed Application Architecture*. O'Reilly Media, Inc., 2000.

[12] B. Benatallah, F. Casati, and F. Toumani, "Web service conversation modeling: a cornerstone for e-business automation," *IEEE Internet Computing*, vol. 8, no. 1, pp. 46–54, 2004.

[13] "What is Client-Server Architecture?," https://www.w3schools.in/what-is-client-server-architecture/. Consultado: 10/04/2021.

[14] "What is a REST API?," https://www.redhat.com/en/topics/api/what-is-a-rest-api. Consultado: 10/04/2021.

[15] "The .NET Compiler Platform SDK," https://docs.microsoft.com/en-us/dotnet/csharp/roslyn-sdk/. Consultado: 10/04/2021.

[16] ".NET 5 vs .NET Framework for server apps," https://docs.microsoft.com/en-us/dotnet/standard/choosing-core-framework-server. Consultado: 10/04/2021.

[17] ".NET Core Roadmap," https://devblogs.microsoft.com/dotnet/net-core-roadmap/. Consultado: 10/04/2021.

[18] "What is the MVC pattern?," https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-5.0. Consultado: 10/04/2021.

Apéndices

A

Detalles de implementación

A.1. Ejemplo de uso de Bootstrap y media queries

A.1.1. Ejemplo de uso de una media query

Código A.1: En este ejemplo estamos haciendo que para tamaños de pantalla menores a 991px de ancho se apliquen las siguientes características.

```
<style>
@media (max-width: 991px){
    html { font-size: 13px;}
    .banner .title { text-align: center;}
    .banner {background-image: none;}
}
</style>
```

A.1.2. Ejemplo de uso del grid de Bootstrap

Código A.2: Código HTML que codifica el comportamiento del grid de bootstrap para el panel de sugerencias

```
<div class="row justify-content-center suggestions-container my-5">
    <div class="col-5 col-md-2 mx-1 mt-4 titles-suggestion">...</div>
    <div class="col-5 col-md-3 col-lg-2 mt-4 suggestion mx-1 mx-md-2">...</div>
    <div class="col-5 col-md-3 col-lg-2 mt-4 suggestion mx-1 mx-md-2">...</div>
    <div class="col-5 col-md-3 col-lg-2 mt-4 suggestion mx-1 mx-md-2">...</div>
</div>
```

A.2. Gestión del token JWT dentro del back-end

Código A.3: Pseudocódigo que muestra cómo se crea y verifica el token

```
1  public string GenerateToken(int minutes){
2   var header = '{"alg":"HS256","typ":"JWT"}'.ToBase64();
3   var data = GenerateData(minutes).ToBase64();
4   var signature = HS256(SECRET_KEY, header+"."+data).ToBase64();
5   return "{header}.{data}.{signature}";
6  }
7  private string GenerateData(int minutes){
8       int secondsInOneMinute = 60;
9       string expDate = DateTime.Now.AddSeconds(minutes*secondsInOneMinute).ToString();
10      return '{"exp":"{expDate}"}';
11 }
12 public bool isTokenValid(string userToken){
13     return IsSignatureValid(userToken) && !IsTimeExpired(userToken);
14 }
15 private bool IsSignatureValid (string userToken) {
16      string signatureFromServer = HS256(SECRET_KEY,
            GetHeader(userToken)+"."+GetData(userToken));
17      signatureFromServer = signatureFromServer.ToBase64();
18      return signatureFromServer == GetSignature(userToken);
19 }
20 private bool IsTimeExpired (string userToken) {
21      return getExpDate(userToken)<DateTime.Now;
22 }
```

A.3. Diagrama UML utilizado para implementar la librería

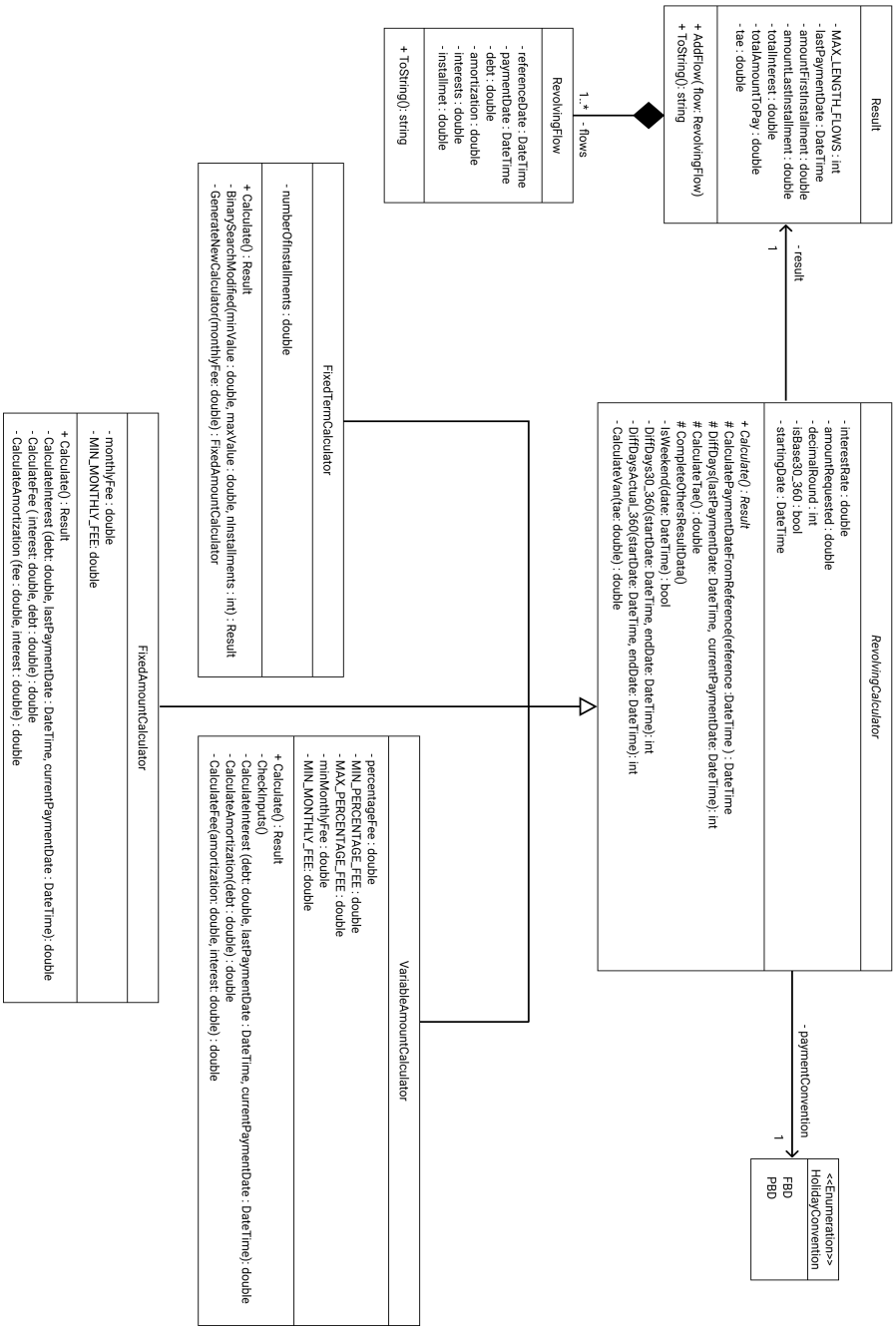

Figura A.1: Diagrama UML seguido durante el desarrollo de la librería. Fuente: elaboración propia.

A.4. Modificación para usar la búsqueda binaria en el cálculo de N plazos

Código A.4: Pseudocódigo que muestra la implementación de la búsqueda binaría para el cálculo con N plazos.

```
1  private Result BinarySearchModified(double minValue, double maxValue, int nInstallments){
2      while(minValue<maxValue){
3          monthlyFee = ((maxValue -minValue ) / 2) + minValue;
4          calculator = GenerateNewCalculator(monthlyFee);
5          result = calculator.Calculate();
6          if(result.Flows.Count == nInstallments){
7              break;
8          }else if(result.flows.Count > nInstallments){
9              minValue = monthlyFee;
10         }else{
11             maxValue = monthlyFee;
12         }
13     }
14     return result;
15 }
```

A.5. Diseño del informe generado por la aplicación web

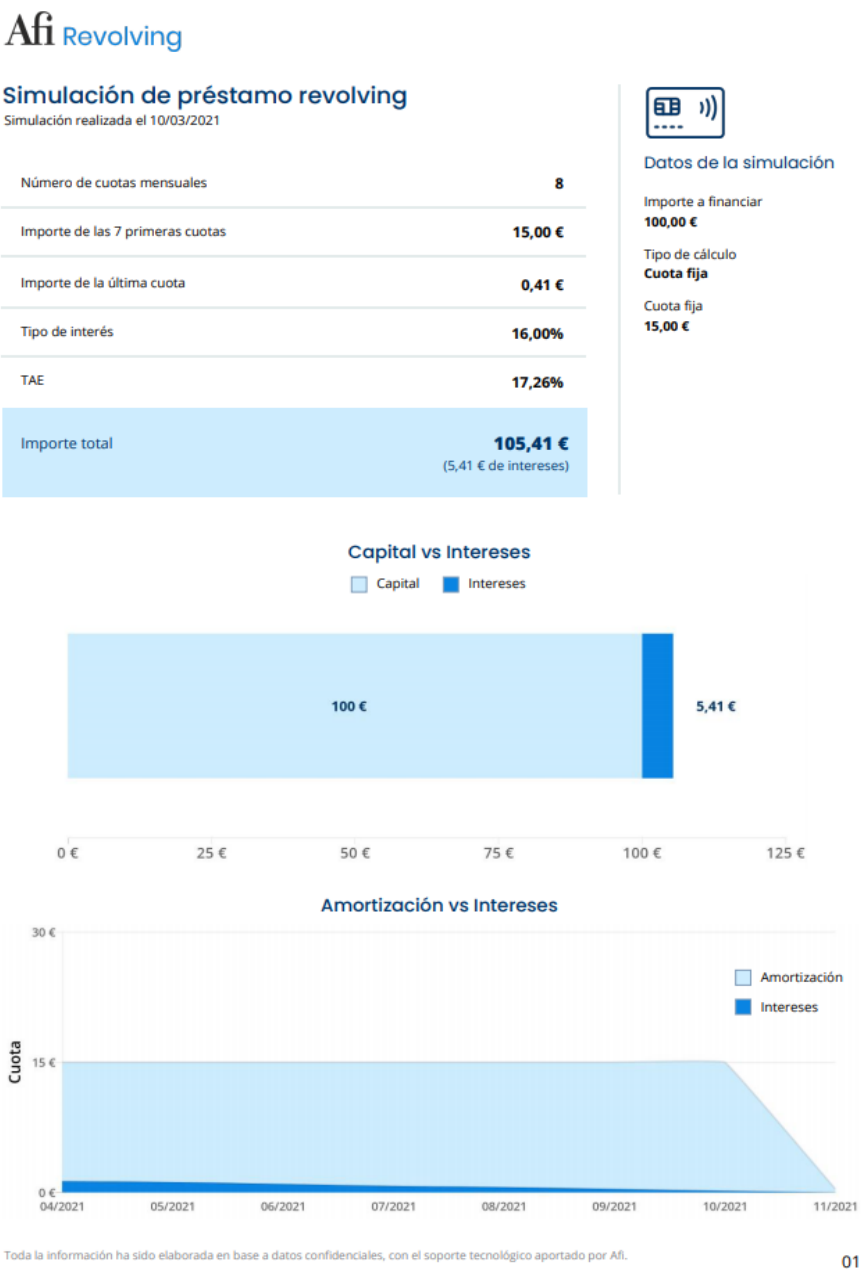

Figura A.2: Informe generado por la aplicación web. Fuente: elaboración propia.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID