Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Desarrollo y Operaciones (DevOps)

Implementación de la cultura DevOps en la empresa

Trabajo fin de estudio presentado por:	Juan José Hernández Alonso
Tipo de trabajo:	Desarrollo práctico
Director/a:	Rafael Merlo Loranca
Fecha:	13 de Julio de 2022

Resumen

El cambio de modelo de negocio de una empresa es uno de los mayores retos que esta puede abordar. La transición desde el mundo de la consultoría al diseño y desarrollo de productos debe darse con pasos firmes y seguros. En ese contexto es donde se encuentra la empresa objetivo y donde se desarrolla este proyecto, consistente en la implementación de la cultura DevOps en Movicoders S.L.

Se ha identificado dicha cultura, sus herramientas y metodologías como la más adecuada para este tipo de cambios y objetivos. Se han analizado desde el entorno específico, algunas de las opciones disponibles, definiendo un *stack* tecnológico adecuado para esta tarea.

Se ha comunicado y acompañado a los miembros del equipo, habilitándoles en el uso de las herramientas Jira, Shell scripting, Jenkins, Docker, AWS y otras como parte de ese *stack*.

Como soporte al cambio, se ha utilizado el proyecto Rehaviva Crece, que ha permitido a los miembros del equipo aplicar esas nuevas técnicas y conocimientos que han adquirido en el proceso. Durante el cual, han aplicado técnicas y metodologías ágiles, alineadas con los principios DevOps que permiten la entrega de valor continuada.

Finalmente, se han evaluado los resultados obtenidos, a nivel técnico y mediante *feedback* con el cliente, frecuencias de entrega, satisfacción del equipo de desarrollo y de Dirección, con un resultado positivo.

Palabras clave: DevOps, Automatización, Medición, Integración continua, Entrega continua

Abstract

Changing a company's business model is one of the biggest challenges this can address. The transition from the world of consulting to the design and development of products must take place with firm and safe steps. In this context is where the target company is located and where this project is developed, consisting of the implementation of the DevOps culture in Movicoders S.L.

This culture, its tools and methodologies have been identified as the most appropriate for this type of change and objectives. Some of the available options have been analyzed from the specific environment, defining a suitable technological stack for this task.

The team members have been communicated and accompanied, enabling them to use Jira, Shell scripting, Jenkins, Docker, AWS and other tools as part of that stack.

As support for change, the Rehaviva Crece project has been used, which has allowed team members to apply these new techniques and knowledge that they have acquired in the process. During which, agile techniques and methodologies have been applied, aligned with the DevOps principles that allow the continuous delivery of value.

Finally, the results obtained have been evaluated, at a technical level and through customer feedback, delivery frequencies, satisfaction of the development and management team, with a positive result.

Keywords: DevOps, Automation, Measurement, Continuous Integration, Continuous Delivery

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En el alcance de este proyecto se aborda el reto tecnológico que supone un cambio de modelo de negocio para una empresa orientada a la consultoría informática hacia desarrollos de producto. Veremos como todos los aspectos y procesos que hasta ahora se venían realizando sufren transformaciones guiadas por la implantación de la cultura DevOps, mucho más ajustada a ese nuevo modelo.

A continuación, se exponen los problemas actuales y algunas de las razones que justifican estos cambios bajo distintos puntos de vista que van desde el tecnológico hasta el económico. Veremos cómo implantando en todas las etapas del ciclo DevOps las herramientas y procesos adecuados, se obtienen unos resultados mucho más cercanos a las pautas que enuncia el manifiesto ágil y que son parte del objetivo último de la empresa.

El objetivo final de este proceso es habilitar a miembros y equipos de trabajo para que sean independientes, maximicen la calidad de su trabajo, su eficiencia y la satisfacción del cliente mediante la implantación de una nueva cultura de trabajo.

1.1.Justificación del trabajo

Movicoders S.L. es una empresa de consultoría TIC que cuenta en su plantilla con unos 40 empleados distribuidos en diferentes áreas (Deployment, SAP, Consultoría de un producto de terceros y Nuevos Desarrollos) y deslocalizados (Madrid, Huesca y Zaragoza). Hasta ahora, el principal modelo de negocio ha sido la consultoría sobre un producto de terceros, pero este está siendo abandonado por la empresa propietaria.

Esto se ha visto por la Dirección como un riesgo, pero también como una oportunidad para cambiar ese modelo de consultoría hacia un modelo de proyectos y de productos. Este cambio está centralizado en el área de Nuevos Desarrollos, donde se está tratando de formar a personal muy joven en nuevas tecnologías y arquitecturas, pero sin que la empresa tenga realmente experiencia en ellas.

El resto de las áreas, permanecerán más estáticas y tienen un perfil de desarrolladores con más experiencia, con un modelo de negocio claro y estable que proporciona el suficiente margen temporal y económico para poder emprender este camino que se expone.

Los miembros del área de Consultoría deberán afrontar una reconversión tecnológica hacia operaciones e infraestructura, o bien incorporándose paulatinamente a otras áreas.

En estas condiciones, el proceso de transformación del modelo de negocio está sufriendo muchos retrasos y pasos atrás motivados por distintos aspectos (rotación de plantilla, errores y correcciones en los desarrollos por falta de experiencia, necesidad de investigación en nuevas tecnologías, pasos a ciegas…). Todas estas condiciones complican el objetivo que se quiere alcanzar y se ha identificado la necesidad de establecer una hoja de ruta bien definida que guíe el proceso de transformación.

1.2. Planteamiento del problema

La cantidad de objetivos específicos que se identifican en este proceso resulta abrumadora ya que no solo se trata de un cambio radical de filosofía, sino que va acompañado de un cambio tecnológico mayúsculo.

El objetivo general que se quiere alcanzar es el cambio de modelo de negocio mediante la implantación de la cultura, técnicas y herramientas DevOps que permitan garantizar la entrega de valor constante y la viabilidad de los proyectos del área, dotando a sus equipos y miembros de la independencia y habilidades necesarias durante el proceso (Cubo, 2022).

1.3. Estructura de la memoria

En el **siguiente apartado** veremos cuáles son los antecedentes de la empresa, su modelo de trabajo actual y con ese punto de partida, se han identificado los puntos clave que debían cambiar. Tras analizar cuál era la situación y alternativas existentes para cada aspecto y se realiza una justificación de las decisiones tomadas.

En el **punto tres** se identifica el objetivo general y los objetivos específicos que nos ayudan a alcanzarlo. Se establece un alcance adecuado a las limitaciones económicas, técnicas y temporales a las que este trabajo está ajustado. Finalmente, tras una evaluación de las metodologías existentes, se plantea y/o diseña una propuesta que se ajusta a las circunstancias del proyecto.

En el **cuarto apartado** se describe el proceso de desarrollo que ha tenido lugar, abarcando todas las etapas, desde la puesta en marcha hasta el cierre de la implantación DevOps. Finalmente, se presenta una evaluación del trabajo realizado, con los resultados obtenidos y las conclusiones extraídas del proceso, y marcando una línea de trabajo que deberá continuar tras el cierre de este proyecto.

2. Contexto y estado del arte

A continuación, se expone con más detalle el contexto del trabajo que aquí se presenta abordando desde distintos aspectos los pormenores que pueden afectar a su alcance. Al tratarse de un proyecto orientado hacia un cambio metodológico, es importante saber cuál es la situación actual de la empresa y sus procesos con un grado mayor de profundidad.

2.1. Antecedentes y contexto empresarial

Toda la empresa es consciente de la necesidad del cambio y de los problemas mencionados y está comprometida a todos los niveles. El organigrama de la empresa es el que sigue:

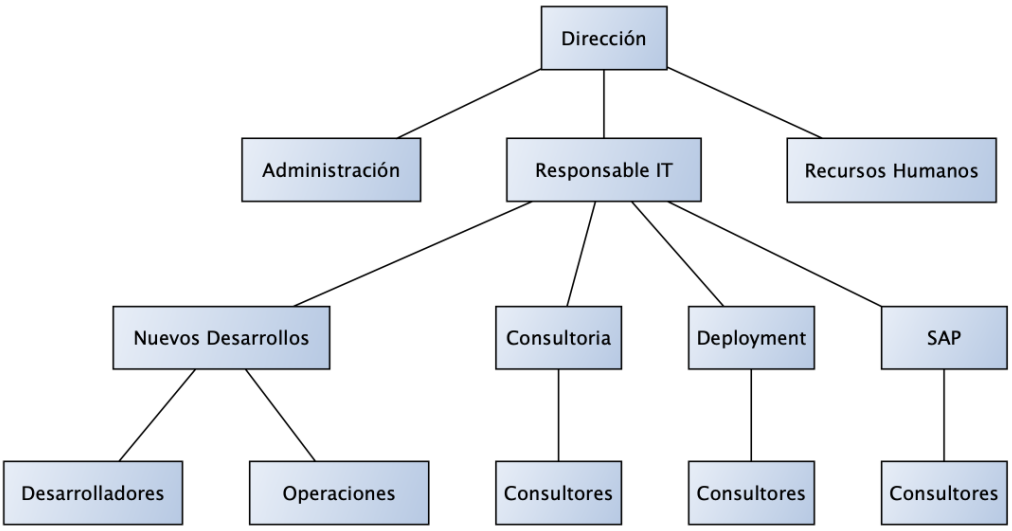


Figura 1. Organización de la empresa Movicoders S.L.

Cada una de las áreas representadas tiene un responsable que, junto con el Responsable IT y Dirección, toman la mayoría de las decisiones organizativas y tecnológicas.

Entre los miembros con un perfil senior, se ha establecido un comité técnico donde se debaten y toman decisiones sobre arquitecturas a utilizar en los proyectos, buenas prácticas y herramientas a incorporar.

Desde la dirección de TI se intenta motivar a todos los empleados. Se entiende que el futuro de la empresa depende de los trabajadores e intenta hacerlos conscientes del porqué de las decisiones tomadas cuando no hacerlos participar en ellas directamente. Algo debatido y consensuado tiene una mayor probabilidad de adopción por los miembros de un colectivo.

La empresa entiende que la presencialidad es clave para acompañar a estos nuevos perfiles y a los trabajadores menos experimentados.

Se fomenta que los responsables de área sean un ejemplo que seguir por los trabajadores a su cargo y una referencia técnica en cómo hacer las cosas.

Se intentan utilizar metodologías y herramientas ágiles a la hora de gestionar los proyectos, aunque es algo en lo que se está trabajando ya que la gestión era habitualmente externa y no había perfiles suficientes capaces de realizar las tareas de Project Manager. Para subsanar esto se está formando al personal con mayor experiencia en los fundamentos de la gestión ágil. Se ha incorporado Jira como herramienta para la gestión, se definen *sprints* y se realizan *dailies* y retrospectivas.

Cuando un nuevo empleado se incorpora a la plantilla, se le proporciona formación en el *stack* tecnológico que se utiliza durante un par de semanas. Además, existen ciertos rituales donde se intenta fomentar la sensación de pertenencia a un grupo, celebrando los hitos personales en equipo.

Los miembros del área de Nuevos Desarrollos son jóvenes entre 18 y 25 años, con poca experiencia y cierta inmadurez técnica y personal. Esto complica la delegación de responsabilidades y añade la necesidad de una revisión y seguimiento cercanos de las tareas desarrolladas. Estas circunstancias no siempre pueden cumplirse por lo desbalanceado de la plantilla actual.

En ocasiones, el tener equipos deslocalizados, complica la comunicación entre miembros del área y dificulta el seguimiento y evolución del personal. Además, probablemente por la costumbre a participar en proyectos de consultoría, algunos desarrolladores senior no tienen toda la implicación deseable en la formación de nuevos miembros.

Actualmente, la empresa está en posición de construir una cultura de producto y es uno de los objetivos innegociables de Dirección.

Se intenta llevar a cabo todos los hitos necesarios para la gestión Agile de proyectos: *kick-off* interno y externo, hay *sprints* de dos semanas, retrospectivas, etc.

Como marco de referencia del estado actual, si revisamos los puntos del manifiesto ágil (Beck, 2001), podemos decir que:

- Se intenta realizar entregas tempranas, aunque no siempre es posible ya sea por las carencias comentadas anteriormente o porque se están desarrollando partes de una arquitectura base para proyectos, que requieren de un esfuerzo extra.
- Se aceptan cambios en los requisitos por parte del cliente.
- Se realizan *sprints* con entregas funcionales.
- Todas las partes (negocio y desarrollo) se implican en el proceso. Conseguir la implicación del cliente no siempre es fácil, pero se intenta que se involucre al máximo.
- Los miembros del equipo están motivados y en un entorno adecuado.
- Se establecen reuniones internas diarias y reuniones externas periódicas donde se comunican avances y bloqueos.
- Se avanza de forma moderada y se producen errores en la entrega por falta de verificación o testeo.
- Se intenta mantener un ritmo de desarrollo continuo.
- Se están incorporando herramientas y técnicas que garanticen una mayor calidad de código a nivel técnico.
- Se está desarrollando una arquitectura y plantillas de componentes que minimicen los esfuerzos.
- Las arquitecturas se someten a debate y, manteniendo la línea inicial, van cambiando, si es necesario.
- Se realizan retrospectivas de proyecto, pero se debería aumentar la frecuencia.

En este aspecto, podemos concluir que, si bien se considera que queda trabajo por hacer, el punto de partida es adecuado y cuenta con el respaldo de todos los niveles de la empresa.

2.2. Antecedentes y contexto tecnológico

Para hablar de DevOps, lo que compone esa filosofía y todo lo que le rodea, se ha establecido un punto de partida. Se han seguido las máximas propuestas por Erich, Amrit y Daneva (2014), donde se indican cuatro palancas para cambio cultural en la empresa:

- Considerar el cambio cultural desde ambas perspectivas: conductor y participante.

- No ser rígido, ni siquiera con los valores DevOps, cuando se dan situaciones excepcionales. Una vez pase la crisis, será el momento de integrar los cambios necesarios.
- Dejar a los equipos decidir qué herramientas utilizar basándose en sus propias habilidades y conocimiento.
- Fomentar la transparencia entre el personal de desarrollo y operaciones.

En otros trabajos ya se ha evaluado la repercusión de esta cultura DevOps en una empresa en pleno proceso de cambio. Es innegable que el cambio de un modelo de consultoría hacia un modelo basado en productos afecta a las organizaciones (Stuckenberg, Fielt y Loser, 2011), y como consecuencia de ese impacto y, gracias a las prácticas DevOps, se aporta un mayor valor y una mayor personalización a los clientes (Hosono y Shimomura, 2012).

De estos puntos, se pueden identificar entre otros conceptos fundamentales e intrínsecos a la cultura DevOps: colaboración, comunicación, automatización, medición y calidad.

Colaboración

Desde el punto de vista de la colaboración, se ha observado que la confianza y responsabilidad de los empleados, difumina las barreras entre desarrollo y operaciones (Feitelson, Frachtenberg y Beck, 2013) reduciendo la necesidad de personal de operaciones y proyectando las habilidades de ambos colectivos como uno solo, más ágil, más flexible.

Automatización

El desarrollo y gestión del software es un proceso creativo, pero también tiene una parte repetitiva. Iteraciones sobre el software, procesos que deben realizarse de forma continua y reiterada… técnicas como el desarrollo dirigido por tests (TDD), scripts y la integración y el despliegue continuos favorecen la repetibilidad y confianza en nuestros procesos. Personas con habilidades y capacidad para automatizar son motores del cambio y factores de éxito en este proceso.

Además, como se menciona en Latorre (2014), estas técnicas como TDD, tienen una gran repercusión en la entrega de valor y satisfacción por parte del cliente, mejorando la calidad de lo entregado y reduciendo periodos de mantenimiento y corrección de errores (otro punto clave del Manifiesto Ágil).

Medición

Uno de los problemas más críticos es la velocidad real de entrega de las metodologías clásicas. Para poder evaluar y corregir desviaciones es necesario medir, tener referencias de lo que el equipo es capaz de realizar en un periodo de tiempo. En palabras de Robin Sharma:

"What gets measured gets improved."

Pero cuando esto se aborda desde equipos con funciones marcadas y separadas (desarrollo contra operaciones), se produce una fricción entre ellos. Mientras los desarrolladores desean entregar su código, operaciones desea garantizar la estabilidad del sistema (Mohamed, 2016).

El hecho de difuminar esas barreras y dotar de un marco de trabajo común a ambos equipos bajo DevOps no solo favorece la gestión, sino la colaboración entre el personal.

Comunicación

Si el objetivo es la colaboración, el medio es la comunicación. Se necesita una comunicación clara, tanto desde el punto de vista del emisor como del receptor del mensaje (en nuestro caso, Desarrollo y Operaciones). Por ello, los procesos y su documentación deben desarrollarse y compartirse bajo un marco común.

Dentro de la gestión del conocimiento en entornos TIC es crucial entender los distintos niveles de información que se requieren a la hora de afrontar un proyecto.

- Desde un punto de vista de investigación y desarrollo, se deben explorar nuevas soluciones disponibles y proponer planes de innovación y mejora.

- Desde un punto de vista analítico, se debe entender qué es realizable y diseñar las especificaciones.
- Desde un punto de vista de implementación, se deben hacer efectivas las especificaciones desde el desarrollo de código, testeo y entrega.

En este último punto se encuentran los equipos de desarrollo y operaciones, que deben compartir ese conocimiento y recursos (Corbin, Dunbar y Zhu, 2007).

2.3. Evaluación y toma de decisiones

Partiendo de la base anteriormente mencionada, y con una clara proyección de mejora e introducción de prácticas DevOps, se han analizado las distintas opciones y herramientas que permiten a la empresa avanzar en varios aspectos.

En este análisis se han tenido en cuenta los siguientes condicionantes como punto de partida:

- La implantación de DevOps en la empresa se intenta abordar minimizando la curva de aprendizaje.
- El equipo objetivo es un equipo joven y con poca experiencia.
- La inversión de tiempo en el cambio está muy condicionada por la carga de trabajo.

2.3.1. Gestión de proyectos

En cuanto a gestión de proyectos, se ha detectado la necesidad de centralizar y estandarizar la gestión de estos. Hasta ahora, la gestión era externalizada y las tareas del personal se limitaban a reportar las horas en Redmine. Con la propiedad de la gestión de proyectos por parte de la empresa, surgen nuevos requisitos que no se llegan a cubrir. Las herramientas evaluadas, que si lo hacen fueron: Excel, Trello, Azure DevOps Board y Jira.

De todas ellas, por experiencia de algunos miembros del equipo en cuanto a la gestión de dichos sistemas se seleccionó Jira. Esta herramienta proporciona las funcionalidades necesarias para soportar la gestión ágil de proyectos, tan ligada a DevOps.

Figura 2. Logotipo Jira.

Como herramientas ofimáticas, la suite Office 365 ya estaba establecida y se ha mantenido su uso, con MS Teams como principal medio de comunicación.

Figura 3. Logotipo Office 365.

2.3.2. Desarrollo

Este aspecto del proyecto se ha visto condicionado por las nuevas arquitecturas, basadas en microservicios, de los productos que se quieren desarrollar. La gestión de código ya estaba establecida con la herramienta Gitea hospedada en un servidor propio. Si bien se evaluó la posibilidad de pasar a otros sistemas alojados en la nube como CodeCommit o Github, la realidad es que, siendo una etapa importante, no se observan razones que motiven la necesidad de cambio inmediato.

Figura 4. Logotipo Gitea.

Un aspecto relevante fue la introducción de herramientas de empaquetamiento en contenedores de nuestras aplicaciones. Si bien existen otras, por simplicidad y características, además de por estandarización, se ha elegido utilizar Docker y archivos Dockerfile.

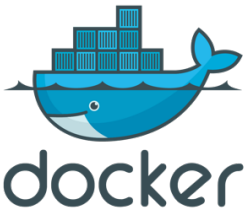


Figura 5. Logotipo Docker.

2.3.3. Integración y entrega continua

El siguiente paso es la integración de los cambios, con sus etapas y pasos de acuerdo con la tecnología. Hay varias soluciones específicas de cada plataforma (AWS Code Build, AWS Code Deploy, VSTS…) pero también existen soluciones transversales como Github Actions, Circle CI, GitLab o Jenkins. En este aspecto no existía experiencia previa por parte de los integrantes del equipo por lo que, tras analizar varias de ellas el equipo se ha decantado por Jenkins, principalmente por ser un estándar de la industria. Si bien hay otras soluciones más sencillas, la cantidad de *plugins*, versatilidad y no estar atados a una plataforma han pesado más.

Figura 6. Logotipo Jenkins.

2.3.4. Despliegues

A la hora de alojar aplicaciones en la nube, hay tres referentes por encima del resto: Amazon, Microsoft y Google. Todos ellos proporcionan soluciones a los requisitos motivados por el cambio emprendido. Como sucede con el punto anterior, el equipo no está familiarizado con ninguna de las nubes mencionadas, por lo que la decisión se ha tomado por la Dirección.

La existencia de un *partner*, experto en AWS, que nos acompañara en esos primeros momentos ha sido crucial en la definición del camino a seguir. Tras el análisis de las herramientas se ha podido concluir que, si bien no es la plataforma más sencilla en cuanto a su manejo, los servicios que ofrecen, la gran documentación existente y que sea realmente el referente del mercado de Clouds apoyan su elección y justifican el esfuerzo de incorporar AWS a nuestro *stack*.

Figura 7. Logotipo Amazon Web Services.

2.3.5. Otros

En cuanto a analítica, se han estudiado dos alternativas hermanas: ElasticSearch y OpenSearch. Por el alcance y profundidad de la transformación, se ha considerado más sencillo el mantenimiento y gestión de AWS OpenSearch, pero el objetivo futuro se ha marcado en utilizar ELK.

Figura 8. Logotipo OpenSearch de AWS.

El motivo de ese objetivo está fundamentado en algunas de las herramientas o *add-ons* existentes que facilitan la monitorización de aplicaciones que esta opción soporta.

Figura 9. Logotipo ElasticSearch, Logstash y Kibana (ELK).

En cuanto a infraestructura como código (IaC), el objetivo es incorporar Terraform en un futuro por ser independiente de las nubes existentes.

Figura 10. Logotipo Terraform.

Para el apoyo al desarrollo y aseguramiento de la calidad, se pondrá el objetivo de incorporar Sonarqube a los pipelines en el futuro.

Figura 11. Logotipo Sonarqube.

Hay otros aspectos que se abordarán como el almacenamiento de imágenes de Docker, librerías, etc. Que se cubrirán con soluciones de AWS como Code Artifact, Elastic Container Registry y similares, pero sin perder de vista soluciones como Nexus, DockerHub, etc.

2.4. Conclusiones del estado del arte

El estado del arte de DevOps está en continua evolución. Una cultura, un movimiento, una filosofía es algo poco tangible, pero con unas raíces fuertes y profundas, en este caso, surgidas de la experiencia. Referentes del mundo del software redactaron el "Manifiesto ágil" que acompaña a DevOps como parte de su naturaleza, pero DevOps no es un proceso, no es un conjunto de reglas y no es la piedra filosofal que nos llevará al éxito de un proyecto. Todo debe evaluarse a la hora de aplicar un punto de vista DevOps, hasta el propio punto de vista.

Hay muchas alternativas, en cuanto a herramientas y metodologías para desarrollar las mismas acciones que se proponen en este proyecto. Elementos que cambiarán, que serán reemplazados y que desaparecerán… por eso lo importante no es el cómo hacer DevOps, es el tener la capacidad de adaptarse al cambio, la disposición a la colaboración y el deseo de comunicación por parte de los integrantes de un equipo de trabajo.

Como conclusión al análisis de la literatura existente, podemos decir que las personas son el elemento central de la cultura DevOps y que seleccionar a los integrantes de un equipo con las habilidades adecuadas será clave para la adopción exitosa de la misma. El cambio debe ser transversal a la empresa y debe entenderse como clave la cooperación, la confianza y la responsabilidad propia y compartida.

La situación ideal para la implantación de estas prácticas son equipos dedicados al desarrollo de un producto, donde se tiene la propiedad de todo el proyecto. Por eso, dado el contexto empresarial, el cambio de filosofía, hace que DevOps sea una elección justificada.

3. Objetivos y metodología de trabajo

En estas condiciones, el proceso de transformación del modelo de negocio está sufriendo muchos retrasos y pasos atrás motivados por distintos aspectos (rotación de plantilla, errores y correcciones en los desarrollos por falta de experiencia, necesidad de investigación en nuevas tecnologías, pasos a ciegas…). Todas estas condiciones complican el objetivo que se quiere alcanzar y se ha identificado la necesidad de establecer una hoja de ruta bien definida que guíe el proceso de transformación.
La cantidad de objetivos específicos que se identifican en este proceso resulta abrumadora ya que no solo se trata de un cambio radical de filosofía, sino que va acompañado de un cambio tecnológico mayúsculo.

Todo proyecto debe tener, de manera recomendable, un único objetivo general claro que de sentido al mismo. La consecución parcial o total del mismo no significa el éxito o el fracaso de este, sino que es el proceso y sus circunstancias las que pueden condicionar ese resultado. En el siguiente apartado se presenta la meta que se ha marcado en el trabajo actual.

3.1. Objetivo general

El objetivo general que se quiere alcanzar es el cambio de modelo de negocio mediante la implantación de la cultura, técnicas y herramientas DevOps que permitan garantizar la entrega de valor constante y la viabilidad de los proyectos del área, dotando a sus equipos y miembros de la independencia y habilidades necesarias durante el proceso.

3.2. Objetivos específicos

Los objetivos más específicos que nos ayudarán a conseguir el objetivo general son:

- Incorporar y fomentar la **cultura** DevOps en la empresa.
- Mejorar la **comunicación** entre los miembros de los equipos.
- Dar a **conocer** la arquitectura, tecnologías y herramientas necesarias.
- Ampliar las **habilidades** técnicas de los desarrolladores.
- Aumentar la **calidad** del código entregable.

- Establecer un nivel técnico mínimo y **uniforme** entre los desarrolladores.
- **Simplificar** el desarrollo estableciendo una base sólida y propia.
- Incorporar TDD en **elementos clave** de nuestros desarrollos.
- Simplificar el proceso de **construcción** del código.
- Minimizar las **dependencias** de terceros.
- Implementar técnicas de ***testing*** automatizado a distintos niveles.
- Diseñar una estrategia de ***releases*** adecuada.
- **Democratizar** el **aprovisionamiento** de infraestructura.
- **Automatizar** despliegues de infraestructura en distintos entornos.
- Establecer estrategias de ***backup***.
- Diseñar un **plan de seguridad** frente a incidentes críticos no cubiertos por la plataforma.
- Añadir **observabilidad** a las aplicaciones e infraestructuras.

De todos estos objetivos se ha realizado un análisis que establece las prioridades de implantación tal como muestra la siguiente tabla:

Tabla 1. Prioridades de implantación de los objetivos específicos.

Objetivo	Prioridad
Incorporar y fomentar la **cultura** DevOps en la empresa.	Alta
Mejorar la **comunicación** entre los miembros de los equipos.	Alta
Dar a **conocer** la arquitectura, tecnologías y herramientas necesarias.	Media
Ampliar las **habilidades** técnicas de los desarrolladores.	Media
Aumentar la **calidad** del código entregable.	Alta
Establecer un nivel técnico mínimo y **uniforme** entre los desarrolladores.	Alta
Simplificar el desarrollo estableciendo una base sólida y propia.	Media
Incorporar TDD en **elementos clave** de nuestros desarrollos.	Media
Simplificar el proceso de **construcción** del código.	Media
Minimizar las **dependencias** de terceros.	Baja

Implementar técnicas de ***testing*** automatizado a distintos niveles.	Alta
Diseñar una estrategia de ***releases*** adecuada.	Media
Democratizar el **aprovisionamiento** de infraestructura.	Baja
Automatizar despliegues de infraestructura en distintos entornos.	Alta
Establecer estrategias de ***backup***.	Baja
Diseñar un **plan de seguridad** frente a incidentes críticos no cubiertos por la plataforma.	Baja
Añadir **observabilidad** a las aplicaciones e infraestructuras.	Media

3.3. Metodología del trabajo

Si planteamos el desarrollo de este alcance mediante una planificación en cascada, nos encontraremos con diferentes retos. Por ejemplo, el número de objetivos específicos planteados y su complejidad hacen surgir un riesgo fundamental en su implementación: la disponibilidad del equipo con el que se van a incorporar estas prácticas y este cambio metodológico.

Al ser objetivos que no tienen una vinculación directa sobre el proyecto subyacente, es difícil establecer un orden de implantación para cada uno de ellos. Por eso, es conveniente que el equipo y el desarrollo de la incorporación de estas metodologías se pueda adaptar a esa variabilidad.

Además, durante este periodo, que se prevé extenso, pueden surgir cambios en la organización (en cuanto a trabajadores y filosofía empresarial), exigencias de resultados tempranos que garanticen el apoyo de dirección, reactividad frente a prioridades de trabajo y necesidades de proyectos contemporáneos. Como se ha mencionado, se entiende que este "meta-proyecto" debe ir acompañado en muchas de sus fases y objetivos de un proyecto de cliente externo que permita focalizar los desarrollos y prácticas en un caso concreto. Esa simultaneidad puede causar que el ciclo de vida del proyecto objetivo de este trabajo se vea afectado en una o varias etapas de su ciclo de vida.

En función de las cargas de trabajo, puede ser necesario que se reorganicen las prioridades o incluso se podría paralizar la implementación de un *skill* o *skills*.

El impacto de muchas de estas casuísticas puede minimizarse con la aplicación de una metodología Agile y aplicando las guías descritas en Schwaber y Sutherland (2020).

El punto de partida de este proyecto es que no es un proyecto software al uso. Si bien define objetivos que se aplican al software desarrollado, no se produce una entrega de un producto concreto. Si evaluamos las circunstancias desde el punto de partida del manifiesto ágil, podemos identificar, sobre los 4 valores principales, las siguientes ventajas:

- **Individuos e interacciones por encima de procesos y herramientas:** El enfoque del proyecto es habilitar con nuevas capacidades a los miembros del equipo, aumentando su independencia y mejorando la comunicación. Se entiende que las personas son el principal valor de la empresa y se dará prioridad a las necesidades de los individuos en su día a día para acelerar la entrega de software de mayor calidad.
- **Software funcionando por encima de documentación exhaustiva:** En este caso, se debería utilizar características en lugar de software. Se priorizará la correcta aplicación de las habilidades adquiridas sobre otras habilidades más tradicionales como la documentación.
- **Colaboración con el cliente por encima de negociación contractual:** Bajo esta perspectiva, cliente y proyecto convergen en los miembros del equipo. Identificar sus necesidades y darles una respuesta temprana y adecuada facilitará la adopción de las soluciones por parte del personal del equipo.
- **Respuesta ante el cambio por encima de seguir un plan:** El enfoque ágil nos permitirá balancear entre los distintos objetivos, según el proyecto en el que se desarrolle. Se irán incorporando *features* o *skills* al equipo de trabajo que se ajusten a las necesidades que el proyecto que soporta este proceso requiera.

Si bien toda la definición del proyecto de cambio de modelo de negocio se ha establecido en una única hoja de ruta, los proyectos y equipos que lo soporten pueden ser variables. Por ello, es importante tener una gestión descentralizada, donde cada equipo sea responsable del objetivo que llevan entre manos. Este, hará que surjan necesidades de cambio que deben ser asumidas y promocionadas por el equipo, intentando que se extienda a otros desarrolladores y proyectos.

Los miembros del equipo deben ser los vehículos del cambio, demostrando que la inversión en esa herramienta o metodología (dependiendo del caso) permite una mejor comunicación, rendimiento y retorno de la inversión.

Si bien el proyecto consiste en la implantación de herramientas y metodologías DevOps, en este punto, además, se analizará cuál sería un caso de uso de esa implantación sobre un proyecto tipo, tal cual se realizaría en la actualidad.

Actualmente, los procesos de consultoría se comercializan de distintas maneras (bolsa de horas o bien precio cerrado). En estos casos, la planificación, si procede, se ha llevado a cabo por parte del cliente. En los nuevos proyectos que se quieren acometer, se han detectado carencias en el equipo que deben corregirse, para obtener un mejor resultado final.

Inicialmente, se debe preparar una base que simplifique y de soporte a las necesidades de los proyectos que se van a realizar. Para ello se ha dedicado tiempo a la implementación de una base de código que acelere la velocidad de desarrollo. Esta librería o librerías irán incorporando elementos nuevos según se requiera en los proyectos de forma incremental y continua.

La arquitectura estándar que define esos proyectos, mayoritariamente basada en microservicios, conlleva la necesidad de realizar desarrollos continuos en paralelo. Teniendo un conjunto base de microservicios y siendo esta similar entre proyectos por la reutilización de componentes, resulta beneficioso que el aprovisionamiento de infraestructura se haga de forma democrática y partiendo de una plantilla estándar. Por ello se ha recomendado usar infraestructura como código que pueda configurarse de acuerdo con las necesidades particulares de cada proyecto.

Muchas veces, el cliente define requisitos durante el periodo de desarrollo, por eso es importante que la comunicación sea rápida y directa. Tanto desde el cliente hacia el equipo de desarrollo como entre los miembros del equipo. Lo mismo sucede con las librerías, arquitecturas… a implementar y utilizar en los desarrollos, deben comunicarse y compartirse para que todo el equipo adquiera la propiedad de este código. Con ello conseguimos un nivel técnico uniforme y mínimo para todos los miembros del equipo.

Esa variabilidad en el alcance puede causar que se vean afectadas partes del desarrollo que se daban por cerradas y que, sin una verificación, han podido cambiar su comportamiento tras

esos cambios. Por ello, se identifica como fundamental, tener una batería de pruebas automatizadas que verifiquen que en el proyecto no se han introducido errores. La idea ha sido incorporar Test Driven Development como parte del proceso de ingeniería (en los procesos donde la incertidumbre en la implementación es alta), lo que supone para el desarrollo un estándar de calidad. Aquellos otros procesos donde TDD no aplica, se ha requerido incorporar, al menos, test unitarios y de integración.

Además, es importante que, tras esos cambios, se verifique la estabilidad del conjunto y reaccionar cuanto antes a posibles fallos que puedan surgir. Por eso se ha recomendado la implementación de un proceso automatizado y continuo de entrega, que no solo pruebe cada componente de forma individual sino todo el conjunto, incluyendo en ese proceso pruebas *end-to-end* de los casos de uso críticos para el proyecto/producto.

Finalmente, se propone añadir a la infraestructura, herramientas que nos permitan la observabilidad de la aplicación y de la plataforma donde están nuestros despliegues. Esto permitirá que el proceso de entrega, puesta en marcha y mantenimiento sea más sencillo y ágil.

Todas estas prácticas, son prácticas alineadas con los principios DevOps que van en la misma línea que los objetivos propuestos en nuestro “meta-proyecto” y que aportan las ventajas descritas a los desarrollos de esta nueva etapa.

4. Desarrollo específico de la contribución

A continuación, se presenta el desarrollo de la implantación de la cultura DevOps en la empresa Movicoders S.L. y las distintas etapas e hitos alcanzados en el mismo.

Puesto que el objeto de este documento debe realizarse de forma paralela como se ha mencionado, la puesta en marcha de cada fase se puede realizar en un proyecto distinto.

Por simplicidad y relevancia, la mayor parte de las técnicas y herramientas se incorporarán al proyecto Rehaviva Crece, que se define a continuación como punto de partida al desarrollo.

En este proyecto se distinguen dos componentes, una plataforma de gestión de actuaciones de rehabilitación de inmuebles que se encuentra en proceso de desarrollo de nuevas funcionalidades, refactorización y mejora (Rehaviva o Rehaviva 2.0) y la ampliación de esa plataforma con el diseño y desarrollo de un sistema de información inteligente (Rehaviva IA), (que tal como se menciona en) está "concebido para la **generación y evaluación del impacto de escenarios de oportunidad de negocio para la rehabilitación de edificios**", (ICCL, 2021).

Rehaviva IA se construye como un **componente independiente, pero a la vez complementario, de la plataforma de gestión Rehaviva**. Se fundamenta sobre el concepto de *open data*, y está orientado al desarrollo de análisis y proyecciones del potencial de rehabilitación del conjunto de los edificios de la localidad objetivo y a los análisis predictivos del comportamiento del mercado de la rehabilitación.

El proyecto cuenta con dos implementaciones piloto en entornos reales de funcionamiento, Zaragoza y Valladolid, que se han considerado representativas por su actividad en rehabilitación de inmuebles y tamaño y que han demostrado y validado el proyecto.

Finalmente, como se menciona en (AEICE, 2022) *"Rehaviva Crece es un **proyecto intersectorial** (construcción y TIC), inter-clúster (AEICE y Tecnara), e interregional (Aragón y Castilla-León)".*

Este proyecto no es parte del estudio como tal, sino parte del contexto en el que se desarrollará el "meta-proyecto" que se expone en este documento. Por ello, no se entrará en detalles innecesarios ni se profundizará en aspectos que no sean relevantes para alcanzar los objetivos anteriormente definidos.

4.1.Planificación / Análisis / Requisitos

Para cada uno de esos proyectos de apoyo, definiremos y se asumirán prácticas bajo el marco SCRUM que se detecten necesarias y resulten útiles. Haciendo que, con el tiempo, se defina una metodología efectiva para el equipo.

Como prefacio se realizarán unas sesiones de descubrimiento de las metodologías ágiles para el equipo. En ellas se comunicarán las responsabilidades, tareas y expectativas que se tienen para el equipo.

En ese momento identificamos inicialmente los roles de SCRUM como:

- **Cliente:** Representantes de Zaragoza Vivienda, ICCL y Ayuntamiento de Valladolid.
- **Product Owner:** Responsable TI.
- **Scrum Master:** Juan José Hernández (autor).
- **Scrum Team:** Desarrolladores frontend (5) y backend (5).

Tras una primera etapa con el cliente, donde se identifiquen los requisitos iniciales, el *product owner* transmitirá al equipo las tareas, creando un *product backlog* de tareas. En ese momento, el *Scrum Master*, junto con la dirección técnica de la empresa, alineará esos objetivos del proyecto con algunos de los aspectos que se quieren incorporar de la cultura DevOps. Ese backlog será estimado y definido

Definiremos el alcance del primer Sprint y las prácticas DevOps a aplicar como tareas transversales al proyecto. Esto nos permitirá mantener el foco de interés del proyecto y el foco de interés global de la empresa. Es muy importante que la experiencia sea satisfactoria y relevante, pudiendo usar así el caso de éxito como motor de cambio para otros equipos e integrantes.

Tanto los eventos SCRUM como las iteraciones se llevarán a cabo de acuerdo con el marco de trabajo, corrigiendo mediante las retrospectivas los pequeños desajustes que se detecten entre el marco y la implantación real.

Se utilizará una primera reunión o ceremonia para la definición del alcance del sprint y, durante todo el desarrollo de este se realizarán reuniones cortas de 15 minutos por MS Teams, que mantendrán al equipo al corriente de los distintos desarrollos realizados en paralelo por los miembros del equipo. Esto aportará transparencia y visibilidad al proyecto.

Al finalizar el sprint se realizará una revisión de lo conseguido y posteriormente, la retrospectiva ya mencionada.

Se ha establecido la duración media de un sprint en 2 semanas de trabajo.

Con el *product backlog* se ha definido la entrega del proyecto: Rehaviva Crece. Este está compuesto de los siguientes requisitos:

Tabla 2. Requisitos de muy alto nivel del proyecto Rehaviva Crece.

Requisito	Esfuerzo
Gestión de usuarios	M
Gestión de maestros	L
Gestión de actuaciones sobre edificios	XL
Sistema de notificaciones	M
Integración con OpenSearch	L
Despliegue de infraestructura automatizada	XL

Con un desglose más detallado en tareas como el siguiente:

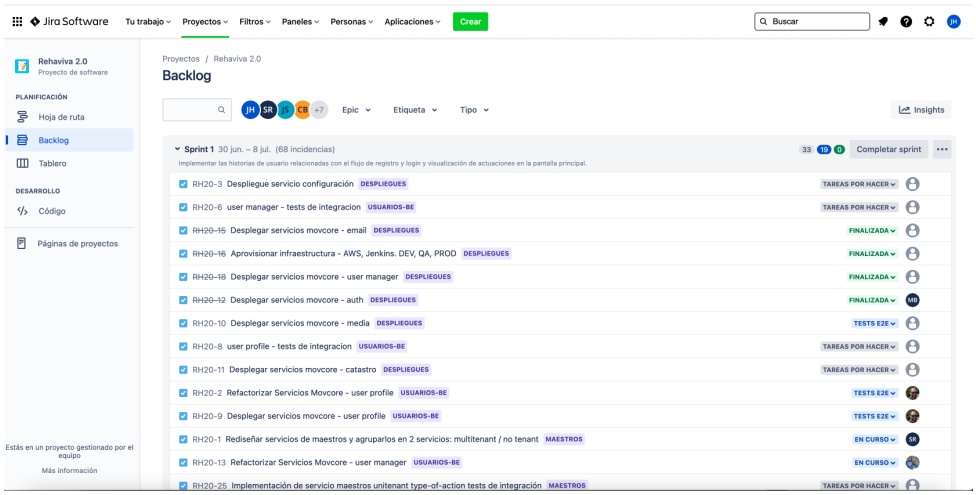


Figura 12. Product Backlog del proyecto Rehaviva Crece.

Se ha realizado una planificación global de los módulos del proyecto sobre los que se incorporarán los distintos objetivos DevOps marcados en el alcance.

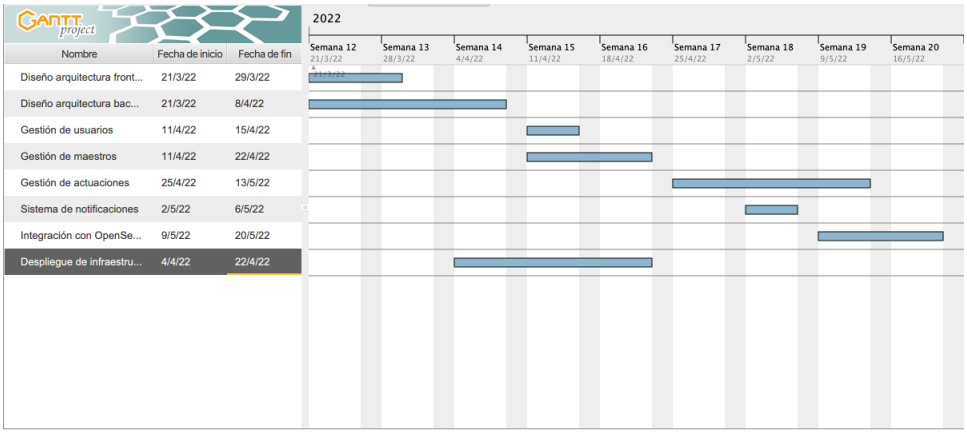


Figura 13. Planificación de los módulos del proyecto Rehaviva Crece.

Con el sprint backlog se ha definido la gestión de usuarios como objetivo del sprint, incluyendo, entre otras, las siguientes tareas:

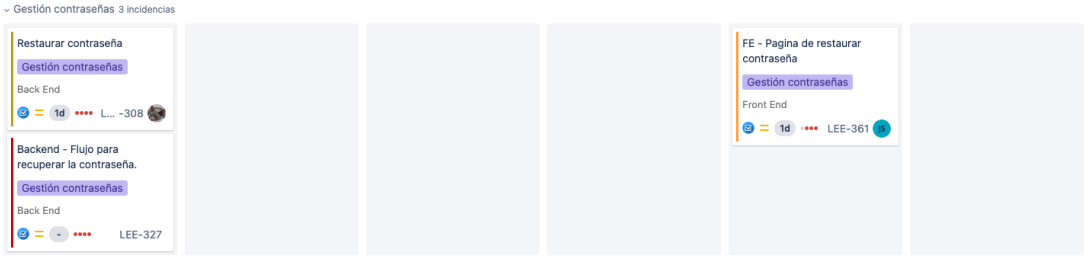


Figura 14. Gestión de contraseñas.

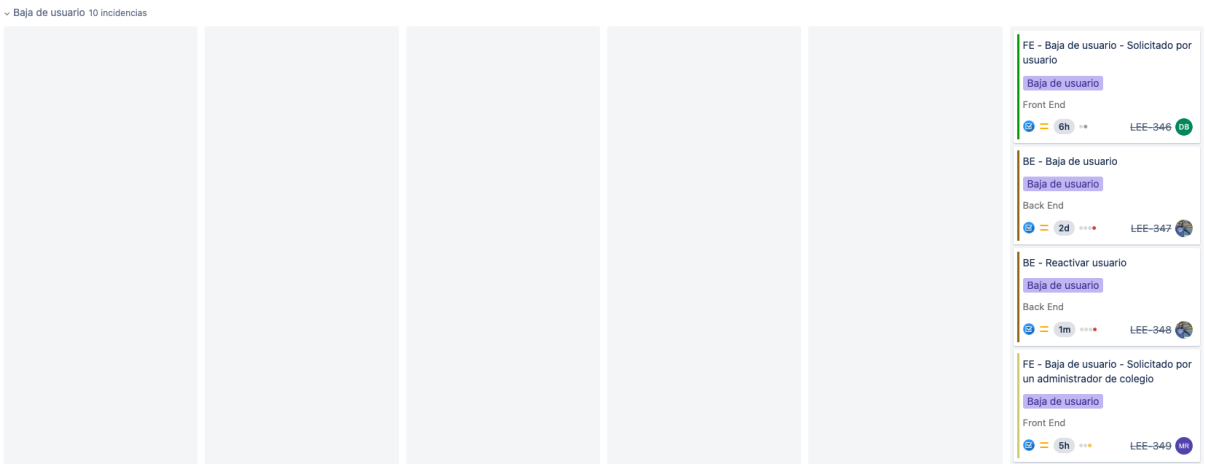


Figura 15. Baja, activación y desactivación de usuarios.

Para llevar a cabo todo esto de una manera organizada, se usará un tablero Kanban que proporcione visibilidad del avance y trabajo del equipo. En él se ha definido el siguiente flujo de trabajo:

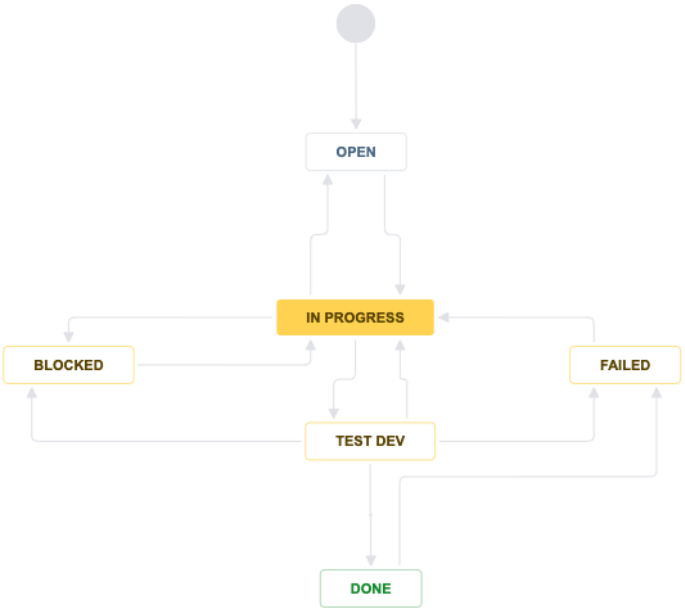


Figura 16. Flujo de trabajo en Kanban.

Finalmente, por la propia naturaleza de los microservicios y gracias a ampliar una visión más allá del alcance del proyecto guía, utilizaremos un enfoque LEAN que nos permita minimizar esfuerzos nuevos de desarrollo a través de la reutilización de microservicios en nuestras aplicaciones.

Todos estos elementos irán muy alineados a los principios y objetivos DevOps que se han planteado en nuestro "meta-proyecto".

4.2. Descripción del sistema desarrollado / Implementación

La idea de Rehaviva Crece es ofrecer el software como servicio (SaaS) a los distintos ayuntamientos/diputaciones que deseen participar. Esto deriva en una serie de condicionantes:

- El sistema debe soportar *multi-tenant* para minimizar los costes de infraestructura sin comprometer la información.
- El sistema debe ser escalable, ya que el número de clientes puede aumentar rápidamente y ser necesario el despliegue de nuevas instancias.
- Los datos de los indicadores pueden ser consumidos a distintos niveles (ayuntamiento, diputación…) lo que requiere de una estructura de búsquedas y cálculos de indicadores específicos que se implementará en un clúster específico.

- Las conexiones con Catastro, son transversales a las instancias necesarias por lo que se separará en un clúster específico y altamente reutilizable.

En las primeras etapas de la planificación, se han definido las arquitecturas a implementar en los dos principales componentes: *backend* y *frontend*.

4.2.1. Backend

En la arquitectura de *backend*, y dadas las anteriores premisas, se ha diseñado la siguiente arquitectura:

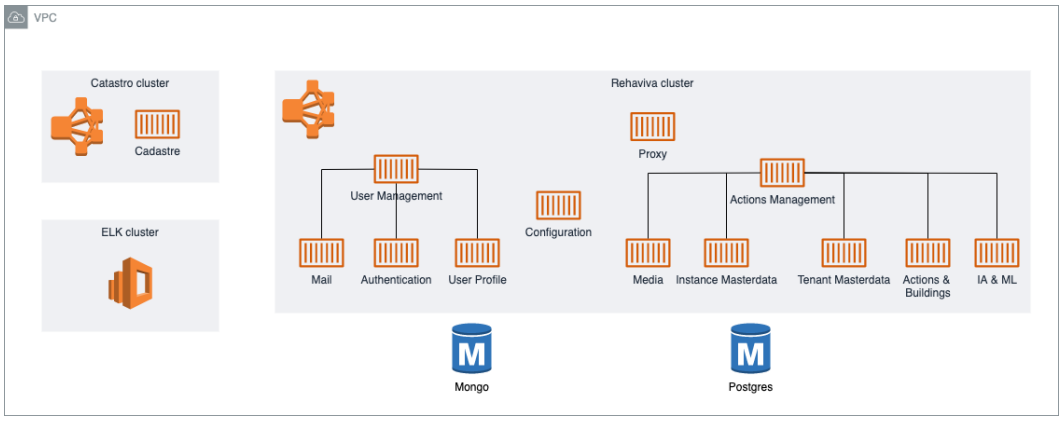


Figura 17. Diseño de arquitectura del componente *backend*.

Se han identificado dos grandes módulos que formarán la aplicación: la gestión de usuarios y la gestión de actuaciones.

Todos los microservicios de esta arquitectura han sido implementados en lenguaje Kotlin con Spring Boot, usando Gradle como gestor de dependencias y constructor de la aplicación. Kotlin es un lenguaje interpretable por la máquina virtual Java (JVM), con un carácter más funcional que Java y una sintaxis más clara y expresiva. Además, Kotlin es totalmente compatible con Java, pudiendo coexistir ambos en un mismo proyecto y es transpilable a otros lenguajes de programación como Javascript. Además, como otros lenguajes modernos como Typescript o Swift, pone barreras al “Billion Dollar Mistake” (Hoare, 2009).

Uno de los principales problemas al abordar el proyecto fue la resolución de aspectos transversales a los distintos microservicios. Para ello, se han desarrollado librerías centradas cada una en uno de ellos como seguridad y control de acceso, *multi-tenancy* de los servicios y utilidades transversales. Estas librerías se desarrollaron sobre una prueba de concepto (PoC) y se alojaron en un repositorio de CodeArtifact.

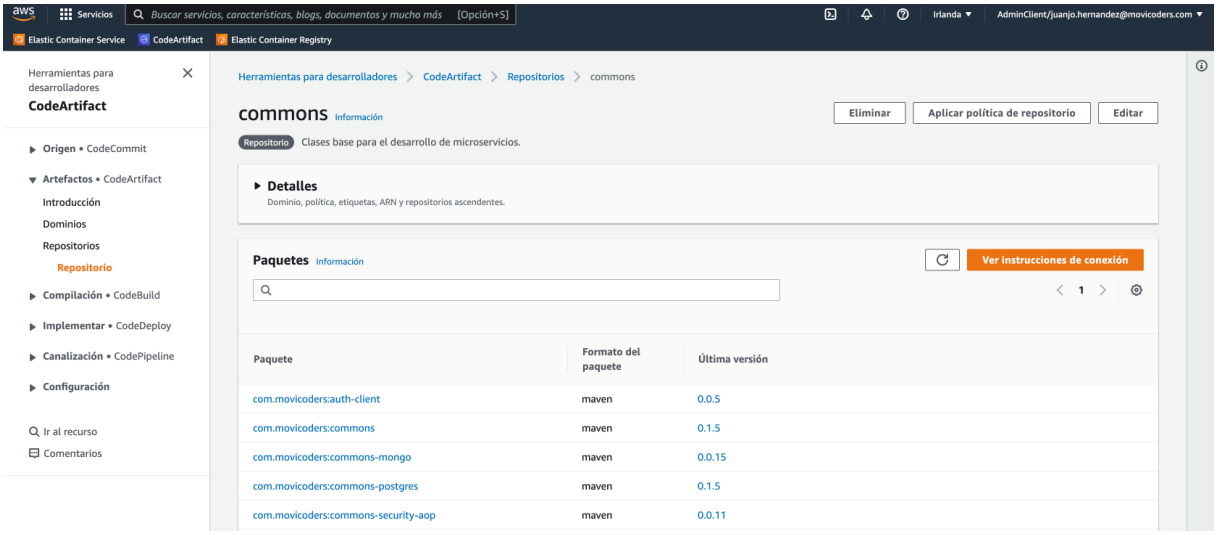


Figura 18. Repositorio CodeArtifact de librerías comunes.

La principal ventaja obtenida es la delegación de la implementación de estos aspectos en un sitio centralizado que garantiza el correcto funcionamiento en todos los microservicios. Para que esto fuese así, se realizó una batería de pruebas sobre la PoC donde se implementó en primera instancia.

El siguiente paso es el empaquetado en contenedores de cada uno de los servicios que, como ya se ha comentado, se realizaría con Docker.

La ventaja de usar la misma tecnología en la inmensa mayoría de contenedores es que muchas de las operaciones pueden estandarizarse. Entre estas se encuentra la definición del Dockerfile que permite, con un solo script ligeramente parametrizable, reutilizarse en todos los elementos.

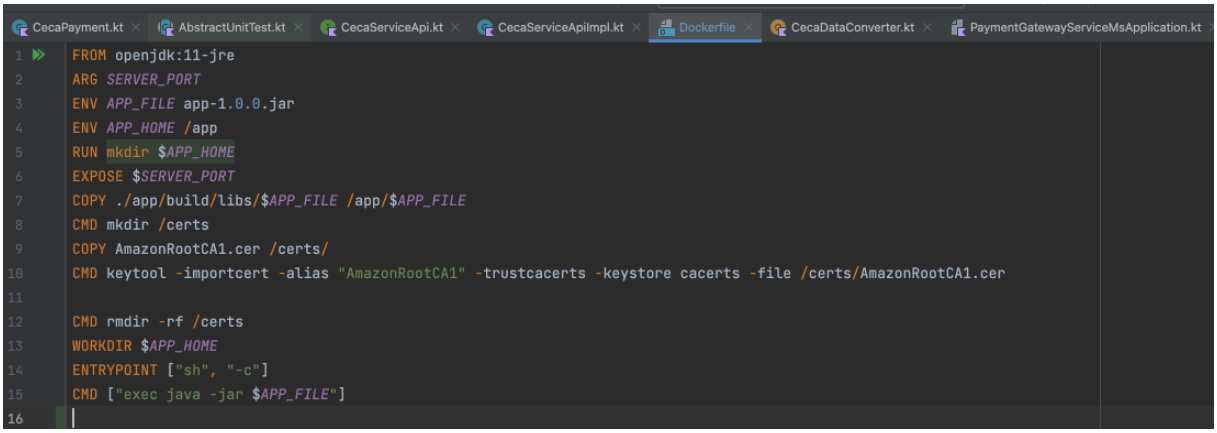


Figura 19. Dockerfile de ejemplo de cada microservicio.

Algo similar sucede con la definición de los pipelines con los archivos Jenkinsfile donde, el primer paso fue identificar las etapas que se querían incorporar al mismo. En nuestro caso, se

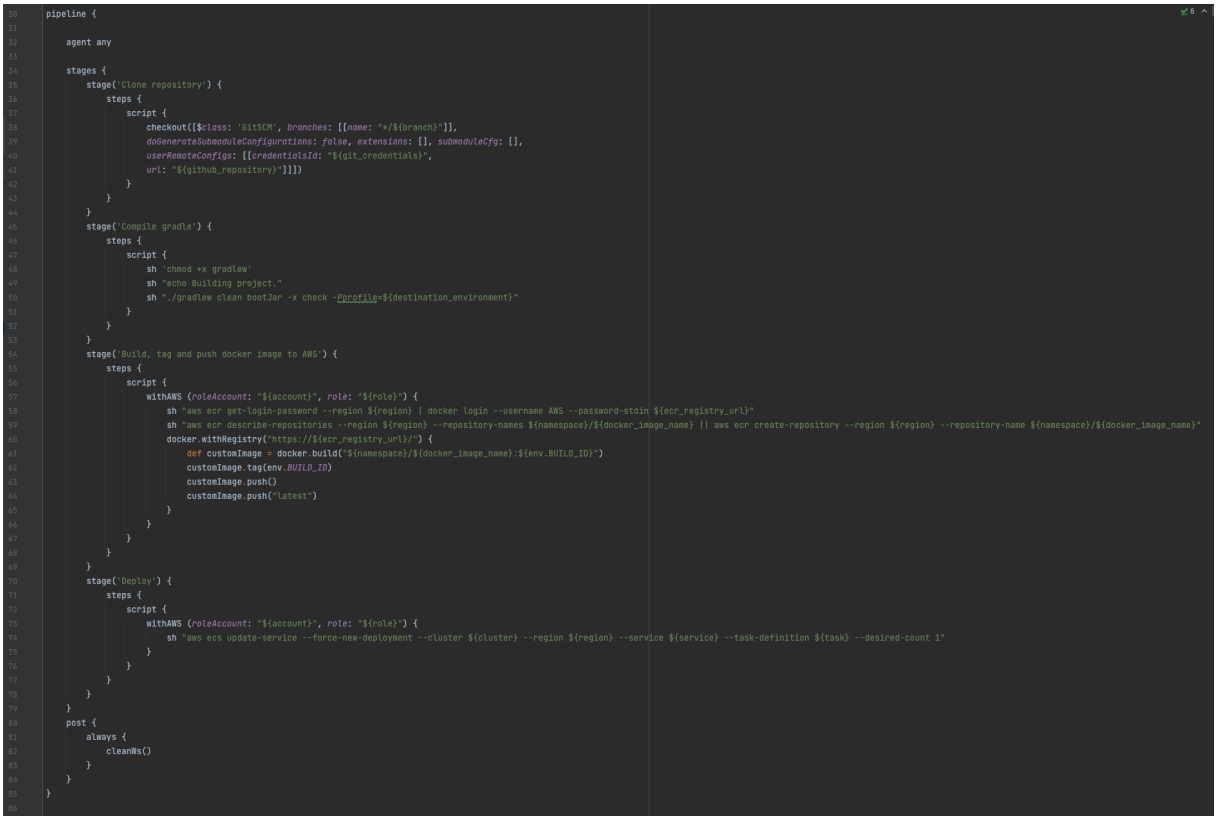
requería clonar desde nuestros repositorios de Gitea el código fuente de cada microservicio en una primera etapa.

Después, se debía compilar el código generando el ejecutable de la aplicación, que, si bien actualmente no lanza los tests, si está previsto que se realice esta verificación en tiempo de compilación.

A continuación, se definió la etapa de dockerización y almacenamiento. En ella el pipeline realiza un *login* sobre ECR, localiza el repositorio y almacena la nueva imagen generada al ejecutar el Dockerfile en el repositorio correspondiente. En el proceso, se añaden etiquetas a esa nueva imagen para que sea localizable.

Un aspecto de mejora identificado en este proceso es que todas las imágenes tienen un tag mutable "*latest*", lo que puede causar confusión en los despliegues cuando se quiera revisar la versión desplegada. Este es uno de los puntos críticos de mejora que se quieren subsanar en el corto plazo, pero dada la escasez de personal disponible con la capacidad de acometer el cambio, se ha pospuesto a una fase posterior.

Finalmente, se definió la etapa de despliegue, donde usando el cliente de AWS, se actualiza el servicio y la tarea asociados al contenedor. Esta etapa es un condicionante de la anterior limitación. A la hora de actualizar la versión de un contenedor en una tarea de ECS, hay que crear una nueva definición de tarea, actualizar la tarea y actualizar el servicio con la nueva versión. Estos pasos suponen una mayor complejidad, que aumentará el tiempo necesario para aplicar la corrección mencionada anteriormente.



```
pipeline {

    agent any

    stages {
        stage('Clone repository') {
            steps {
                script {
                    checkout([$class: 'GitSCM', branches: [[name: "*/${branch}"]],
                    doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],
                    userRemoteConfigs: [[credentialsId: "${git_credentials}",
                    url: "${github_repository}"]]])
                }
            }
        }
        stage('Compile gradle') {
            steps {
                script {
                    sh 'chmod +x gradlew'
                    sh "echo Building project."
                    sh "./gradlew clean bootJar -x check -Pprofile=${destination_environment}"
                }
            }
        }
        stage('Build, tag and push docker image to AWS') {
            steps {
                script {
                    withAWS (roleAccount: "${account}", role: "${role}") {
                        sh "aws ecr get-login-password --region ${region} | docker login --username AWS --password-stdin ${ecr_registry_url}"
                        sh "aws ecr describe-repositories --region ${region} --repository-names ${namespace}/${docker_image_name} || aws ecr create-repository --region ${region} --repository-name ${namespace}/${docker_image_name}"
                        docker.withRegistry("https://${ecr_registry_url}/") {
                            def customImage = docker.build("${namespace}/${docker_image_name}:${env.BUILD_ID}")
                            customImage.tag(env.BUILD_ID)
                            customImage.push()
                            customImage.push("latest")
                        }
                    }
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    withAWS (roleAccount: "${account}", role: "${role}") {
                        sh "aws ecs update-service --force-new-deployment --cluster ${cluster} --region ${region} --service ${service} --task-definition ${task} --desired-count 1"
                    }
                }
            }
        }
    }
    post {
        always {
            cleanWs()
        }
    }
}
```

Figura 20. Definición del pipeline en Jenkinsfile.

La principal característica de estos pipelines es la parametrización de todos sus valores para poder mantener en paralelo distintos entornos (en este caso: Desarrollo y Producción) mediante el mismo pipeline, solamente cambiando un parámetro asociado a la rama que hace el despliegue. En la imagen anterior, se ha omitido la definición de variables ya que contiene datos de configuración sensibles, pero si se pueden observar los nombres de las variables utilizadas a lo largo de los comandos.

Una vez implementado el pipeline podemos observar cómo todos esos pasos se corresponden con las siguientes imágenes. Comenzando por la ejecución de los pipelines en Jenkins:

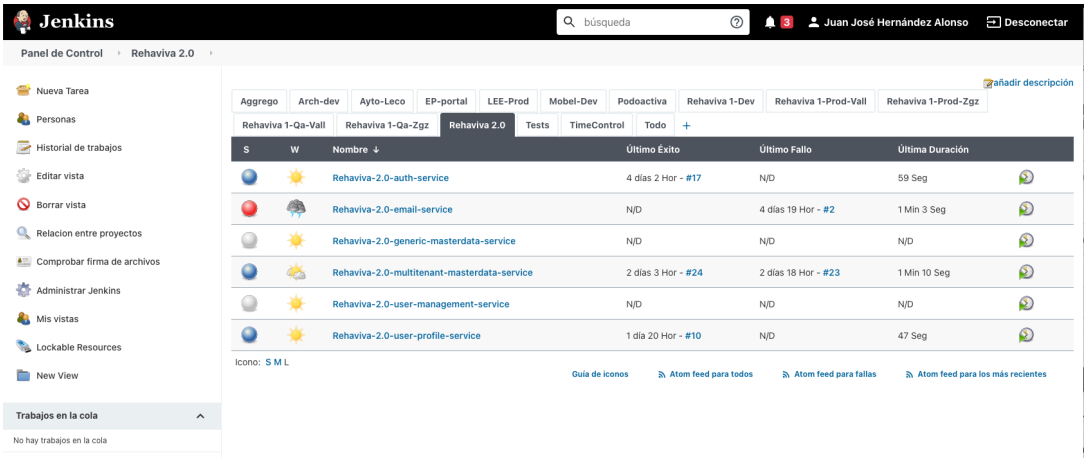


Figura 21. Despliegues en Jenkins del proyecto Rehaviva Crece.

Donde, durante la ejecución satisfactoria se genera y registra una imagen Docker en ECR:

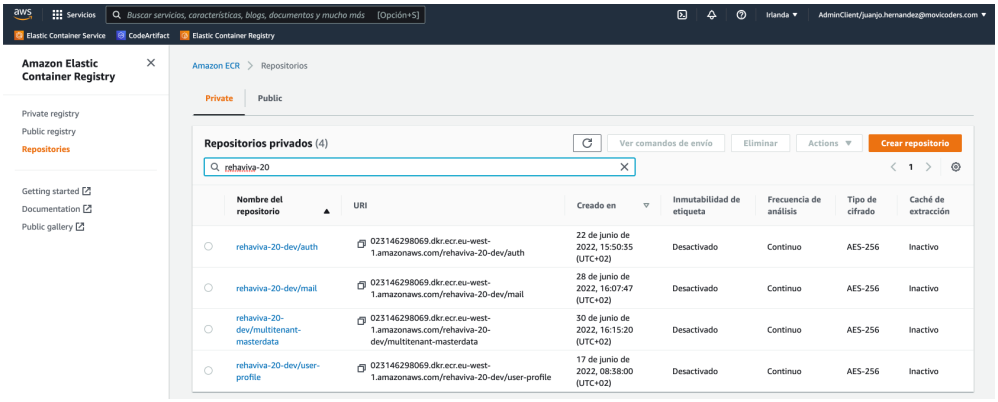


Figura 22. Repositorio ECR del proyecto Rehaviva Crece.

Y, tras ese registro, se despliega la nueva imagen en el clúster del entorno correspondiente a la rama que lanzó el pipeline.

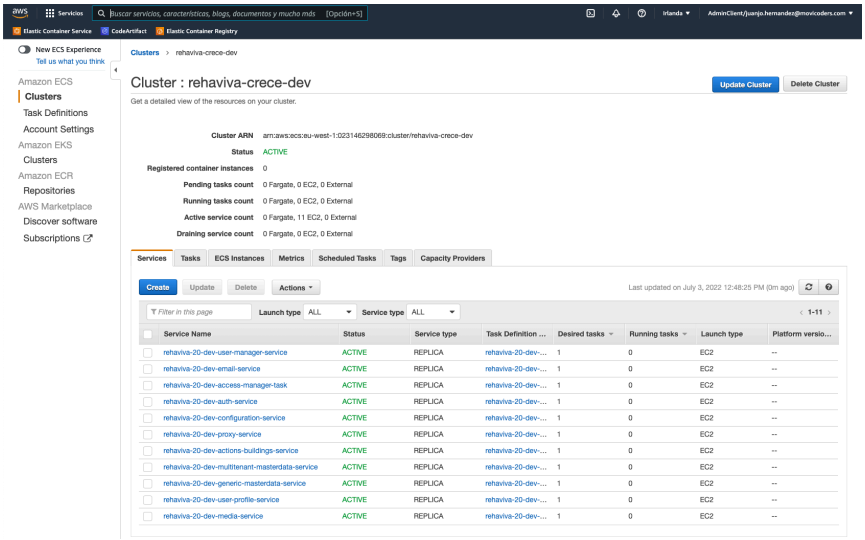


Figura 23. Clúster ECS del entorno de desarrollo del proyecto Rehaviva Crece.

4.2.2. Frontend

En el caso de *frontend*, se ha seguido una estrategia similar a *backend*. La tecnología utilizada ha sido ReactJS con Typescript. El principal motivo es el conocimiento y uso de esta dentro de la empresa. En este caso, la distribución de la aplicación también se hace en un único contenedor, aunque se ha sopesado la opción de utilizar S3 y CDN (servicios de almacenamiento y distribución de AWS), para reducir los tiempos de ejecución del pipeline.

Para la definición de la arquitectura de *frontend*, se ha contratado a un consultor externo que ha impartido una serie de talleres y sesiones formativas en formato Dojo. En este, los miembros del equipo participaban de la definición de la arquitectura y realizaban durante el resto de la jornada avances en la misma, pero ajustándose a las características del proyecto Rehaviva Crece en desarrollo.

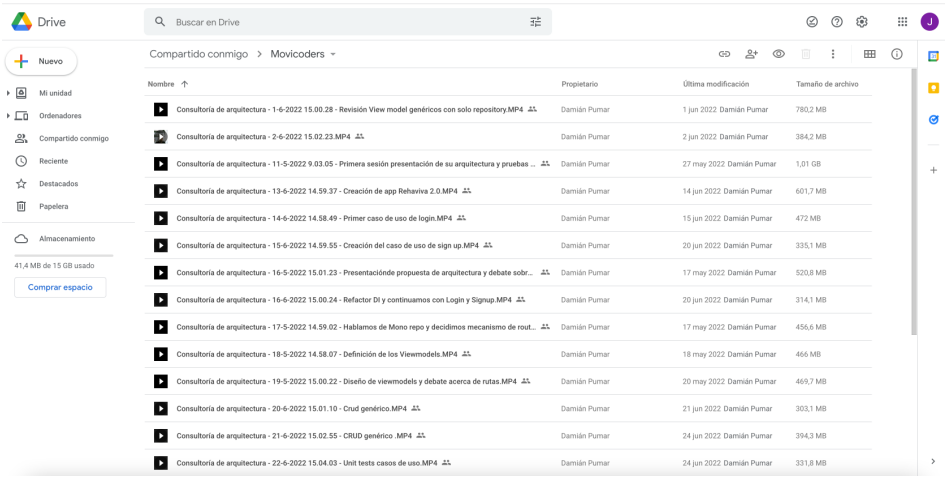


Figura 24. Grabaciones de las sesiones Mob-Programming de definición de arquitectura frontend.

Dicha arquitectura se ha diseñado a partir de unas bases sólidas de la industria como son Clean Architecture y Arquitectura Hexagonal.

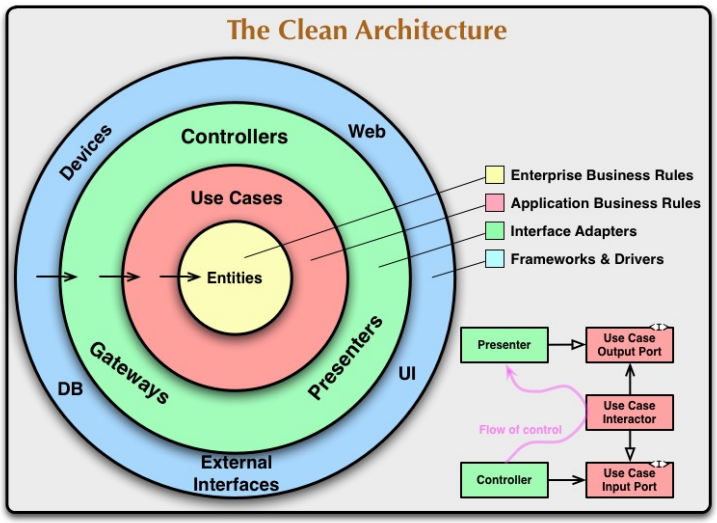


Figura 25. Diagrama *Clean Architecture*.

En este caso, se ha habilitado un nuevo repositorio de CodeArtifact para almacenar las librerías correspondientes a cada una de las capas de la arquitectura definida.

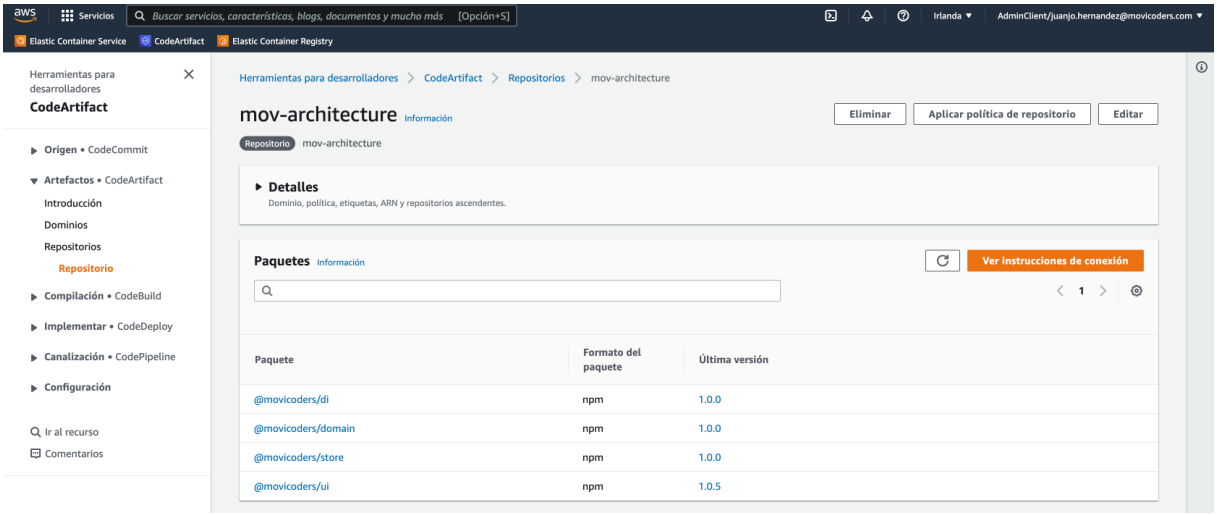


Figura 26. Paquetes npm de la arquitectura *frontend*.

En el siguiente paso, empaquetar en contenedores, se ha realizado de igual manera en un Dockerfile.

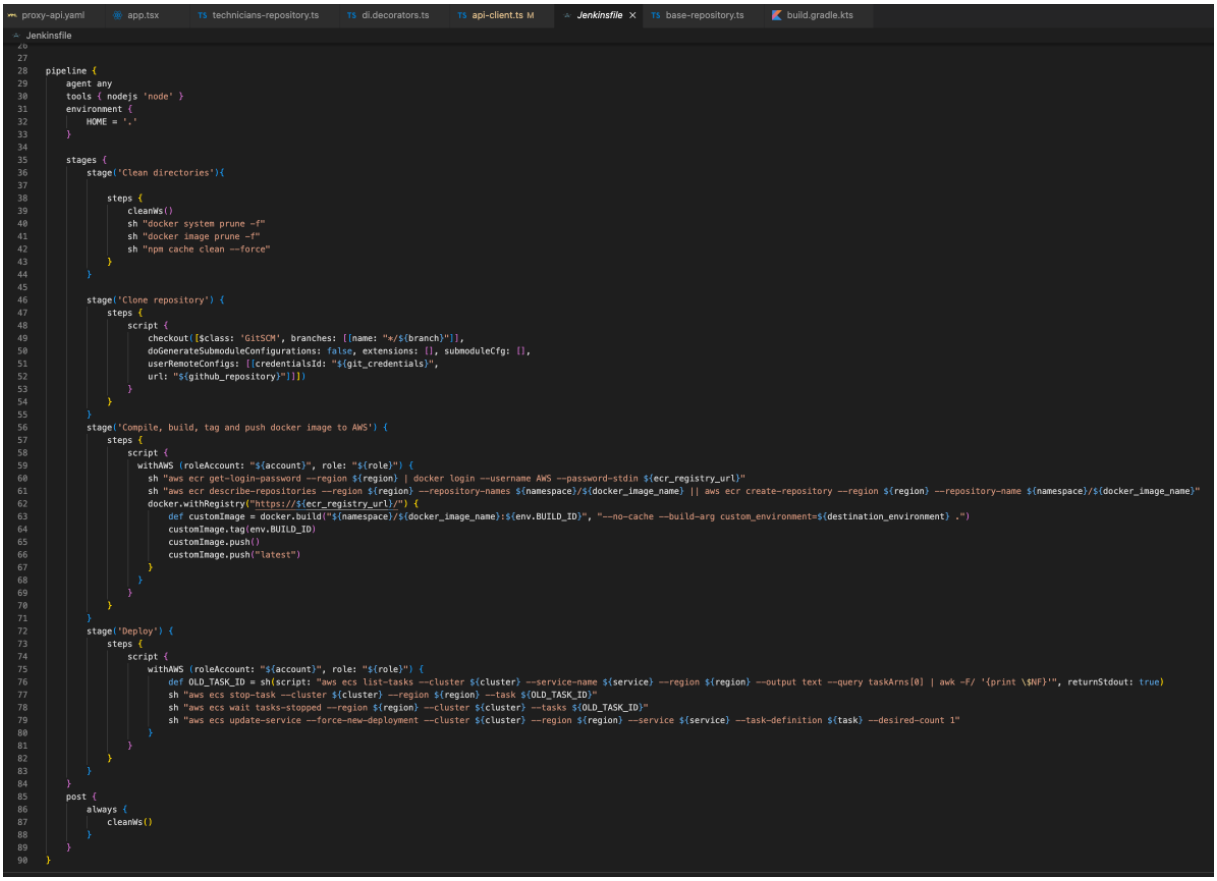
```
FROM node:14-alpine as builder
ARG custom_environment
WORKDIR '/builddir'
COPY . .
RUN mkdir ~/.aws
RUN echo -e "[default]\naws_access_key_id =[redacted]" > ~/.aws/credentials
RUN cat ~/.aws/credentials
RUN npm config set unsafe-perm true
RUN apk update && \
  apk upgrade && \
  apk add git python3 && \
  apk add py-pip curl && \
  pip install awscli && \
  apk --no-cache add --virtual native-deps \
  g++ gcc libgcc libstdc++ linux-headers make
RUN sh aws_login.sh
RUN npm install --quiet node-gyp -g &&\
  yarn install && \
  apk del native-deps      Juan José Hernández Alonso, 5 months ago • master: Adding jenkinsfile and dockerfile …
RUN aws codeartifact login --tool npm --region eu-west-1 --repository mov-components --domain movicoders-libs --domain-owner 023146298069
RUN env
RUN echo $custom_environment
RUN yarn && npm rebuild node-sass && yarn run build_$custom_environment && ls -al build

FROM nginx:alpine
COPY --from=builder /builddir/build /usr/share/nginx/html
RUN rm /etc/nginx/conf.d/default.conf
COPY --from=builder /builddir/nginx.conf /etc/nginx/conf.d/default.conf
ENV NODE_ENV=production
#EXPOSE 3000
#CMD ["npm", "start"]
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

Figura 27. Dockerfile proyecto *frontend*.

En este caso existía el problema de que a esta tecnología le cuesta mucho tiempo realizar el proceso. Actualmente la ejecución de un *pipeline* en *backend* no supera los dos minutos, mientras que en *frontend* está disparado hasta los trece minutos. Esto conlleva unos tiempos muertos para los desarrolladores que, si el pipeline no finaliza correctamente, se pueden ver incrementados de forma exponencial.

Figura 28. Ejecución de un pipeline para *frontend*.

Como se puede observar en la ejecución, las etapas de nuestro pipeline son muy similares a las de *backend*, y de igual manera han supuesto un estándar para todos los proyectos *frontend* en esta tecnología. Están parametrizadas de igual manera y el despliegue se hace utilizando una imagen Docker que es actualizada en la tarea relacionada con el servicio *frontend* en ECS.



```
pipeline {
    agent any
    tools { nodejs 'node' }
    environment {
        HOME = '.'
    }

    stages {
        stage('Clean directories'){

            steps {
                cleanWs()
                sh "docker system prune -f"
                sh "docker image prune -f"
                sh "npm cache clean --force"
            }
        }

        stage('Clone repository') {
            steps {
                script {
                    checkout([$class: 'GitSCM', branches: [[name: "*/${branch}"]],
                    doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [],
                    userRemoteConfigs: [[credentialsId: "${git_credentials}",
                    url: "${github_repository}"]]])
                }
            }
        }
        stage('Compile, build, tag and push docker image to AWS') {
            steps {
                script {
                  withAWS (roleAccount: "${account}", role: "${role}") {
                    sh "aws ecr get-login-password --region ${region} | docker login --username AWS --password-stdin ${ecr_registry_url}"
                    sh "aws ecr describe-repositories --region ${region} --repository-names ${namespace}/${docker_image_name} || aws ecr create-repository --region ${region} --repository-name ${namespace}/${docker_image_name}"
                    docker.withRegistry("https://${ecr_registry_url}/") {
                        def customImage = docker.build("${namespace}/${docker_image_name}:${env.BUILD_ID}", "--no-cache --build-arg custom_environment=${destination_environment} .")
                        customImage.tag(env.BUILD_ID)
                        customImage.push()
                        customImage.push("latest")
                    }
                  }
                }
            }
        }
        stage('Deploy') {
            steps {
                script {
                    withAWS (roleAccount: "${account}", role: "${role}") {
                        def OLD_TASK_ID = sh(script: "aws ecs list-tasks --cluster ${cluster} --service-name ${service} --region ${region} --output text --query taskArns[0] | awk -F/ '{print \$NF}'", returnStdout: true)
                        sh "aws ecs stop-task --cluster ${cluster} --region ${region} --task ${OLD_TASK_ID}"
                        sh "aws ecs wait tasks-stopped --region ${region} --cluster ${cluster} --tasks ${OLD_TASK_ID}"
                        sh "aws ecs update-service --force-new-deployment --cluster ${cluster} --region ${region} --service ${service} --task-definition ${task} --desired-count 1"
                    }
                }
            }
        }
    }
    post {
        always {
            cleanWs()
        }
    }
}
```

Figura 29. Pipeline de un proyecto *frontend*.

El resultado del despliegue en AWS ECS es inmediato, pero como sucede en *backend*, hay que mejorar la estrategia de *releases* que se utiliza actualmente.

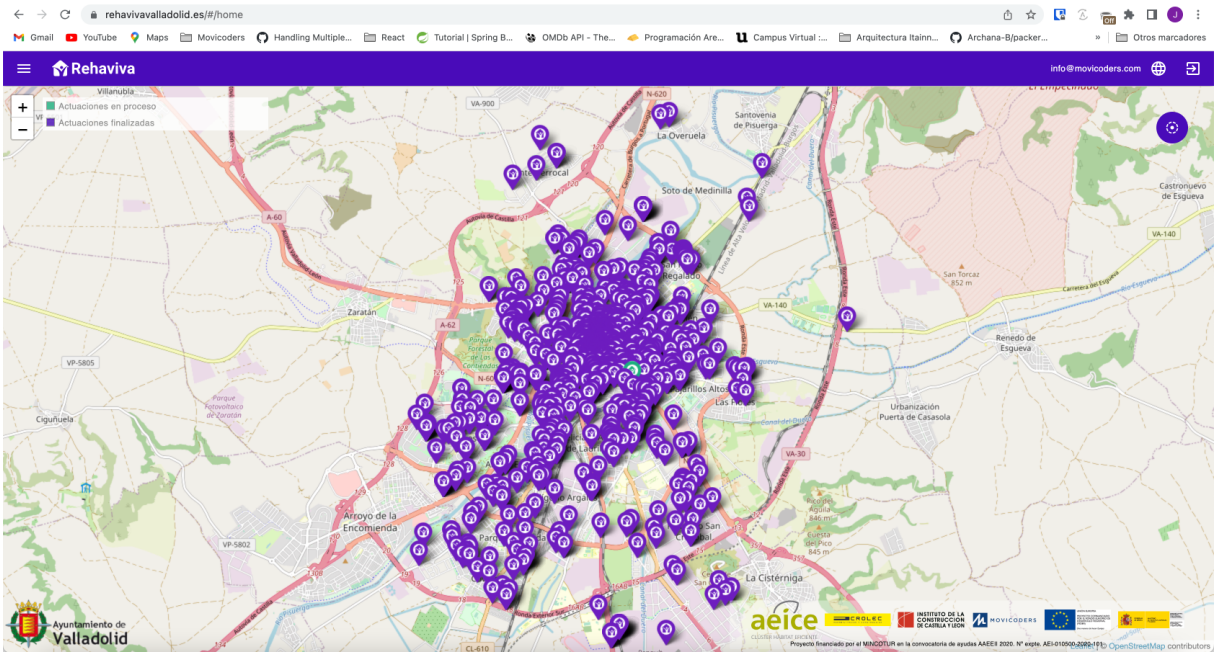


Figura 30. Proyecto Rehaviva Crece en producción.

4.2.3. Código, infraestructura y otros componentes

En cuanto al código, todos los microservicios han incorporado tests automatizados y de integración, utilizando *test containers*, lo que ha minimizado el número de errores y ha llegado a convertirse en un estándar de desarrollo dentro de la empresa.

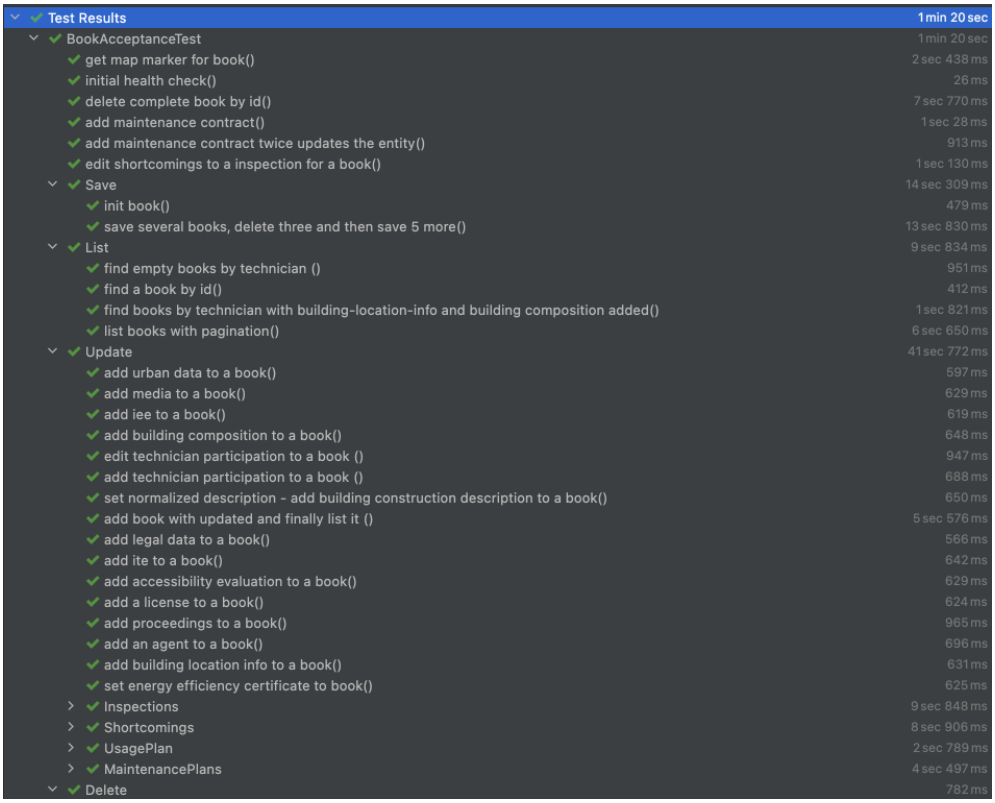


Figura 31. Ejecución de los tests de aceptación de un microservicio.

Lo mismo está sucediendo con los tests unitarios y *end-to-end* que se empiezan a llevar a cabo en los proyectos *frontend*.

La infraestructura propia del proyecto Rehaviva Crece se ha diseñado siguiendo el esquema que se muestra a continuación:

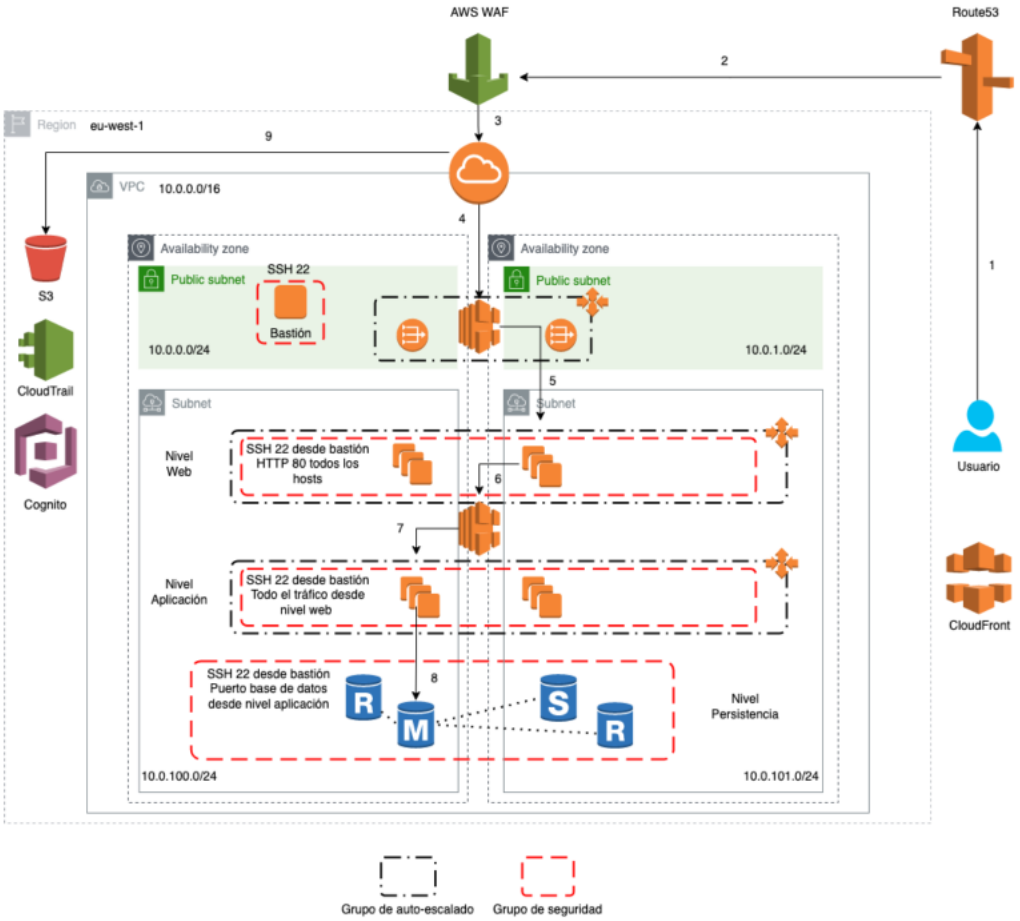


Figura 32. Arquitectura Rehaviva Crece.

Para dar soporte a estos desarrollos se ha tenido que diseñar, implementar y mantener la infraestructura necesaria para llevarlos a cabo.

Tras una primera etapa de formación, donde ningún miembro del equipo tenía experiencia previa, se han implementado políticas y niveles de acceso a los distintos servicios de AWS gracias a IAM.

Para la ejecución de los pipelines se ha montado una instancia de EC2 con Jenkins, que se ha instalado y configurado desde cero. Desde el momento de montaje hasta el momento actual, hay más de cien pipelines correspondientes a una decena de entornos.

Se han configurado los repositorios de ECR y CodeArtifact para una mejor gestión de los entregables y una mayor reutilización de los recursos. Prueba de ello es que muchos de los microservicios y todas las librerías ya se están usando en otros proyectos.

Todos los servicios incorporan logs de CloudWatch para una mejor depuración de errores y revisión de los procesos.

Al utilizar microservicios que se comunican usando el protocolo HTTP, se ha definido como estándar el uso de OpenApi para la definición de las API y se ha creado una herramienta para la generación y publicación de clientes. Esto permite actualizar las llamadas usadas entre microservicios con unos cambios mínimos y menor riesgo de fallo.

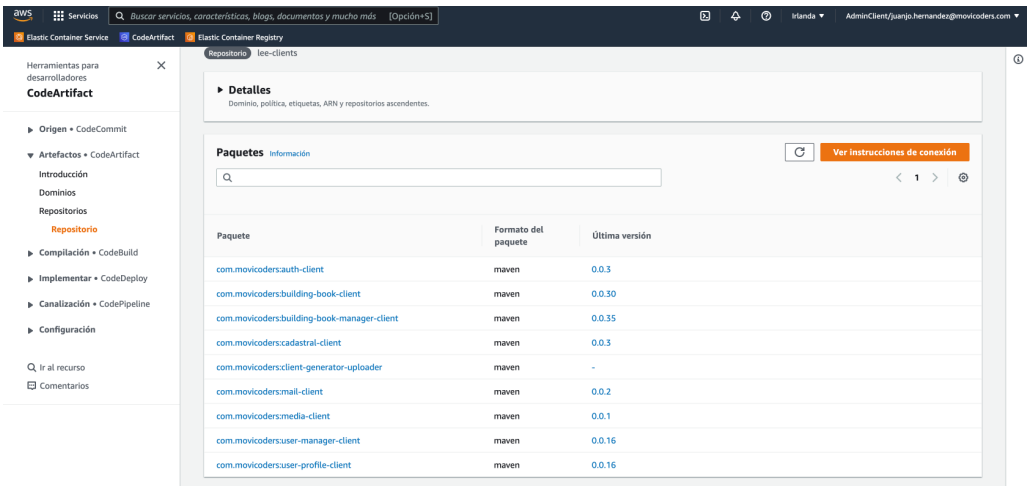


Figura 33. Clientes para comunicaciones HTTP entre microservicios.

Finalmente, se ha creado una plantilla y una herramienta para la autogeneración de microservicios con una implementación de las operaciones básicas más comunes en nuestros proyectos. Para hacer dicha herramienta independiente de la plataforma, se ha utilizado Docker Compose y un *script bash* que, mediante los parámetros proporcionados en la descripción genera una implementación que permite acelerar el desarrollo de nuevos servicios enormemente.

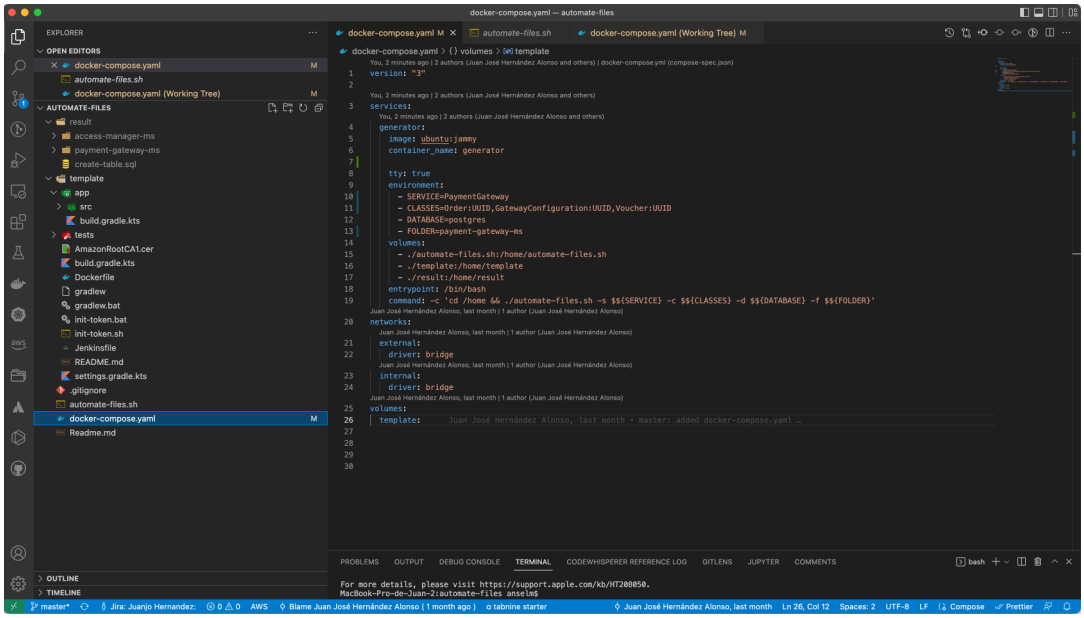


Figura 34. Proyecto para la generación de microservicios.

4.3. Evaluación

Como sucede con el proceso de desarrollo e implantación que define este proyecto, la evaluación es un proceso complejo, ya que no hay un resultado directamente tangible, como puede ser un producto en producción o un entregable. Sin embargo, sí que se pueden evaluar las distintas acciones que se han llevado y algunos de los resultados obtenidos en este proceso.

4.3.1. Formación

Para evaluar las acciones formativas realizadas por personal externo sobre el equipo de *frontend*, se ha realizado una encuesta a cada uno de los miembros con las siguientes preguntas:

Tabla 3. Formato de evaluación de la formación.

Pregunta	Valoración	Comentarios
Formato elegido para impartir la formación/acompañamiento en general	1-5	
Formador y sus conocimientos	1-5	
Interés personal en acudir a esta formación	1-5	
Implicación personal en la formación	1-5	
Aprovechamiento de la formación en un futuro	1-5	
Aplicabilidad a corto plazo de los conocimientos impartidos	1-5	
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	1-5	
Esfuerzo personal extra para profundizar en la formación	1-5	
Nivel de aprovechamiento del curso	1-5	

Idoneidad del planteamiento del curso	1-5	
Esfuerzo realizado por la empresa para proporcionar este curso	1-5	
Valoración del curso en general	1-5	

Los resultados obtenidos en esa encuesta han sido los siguientes:

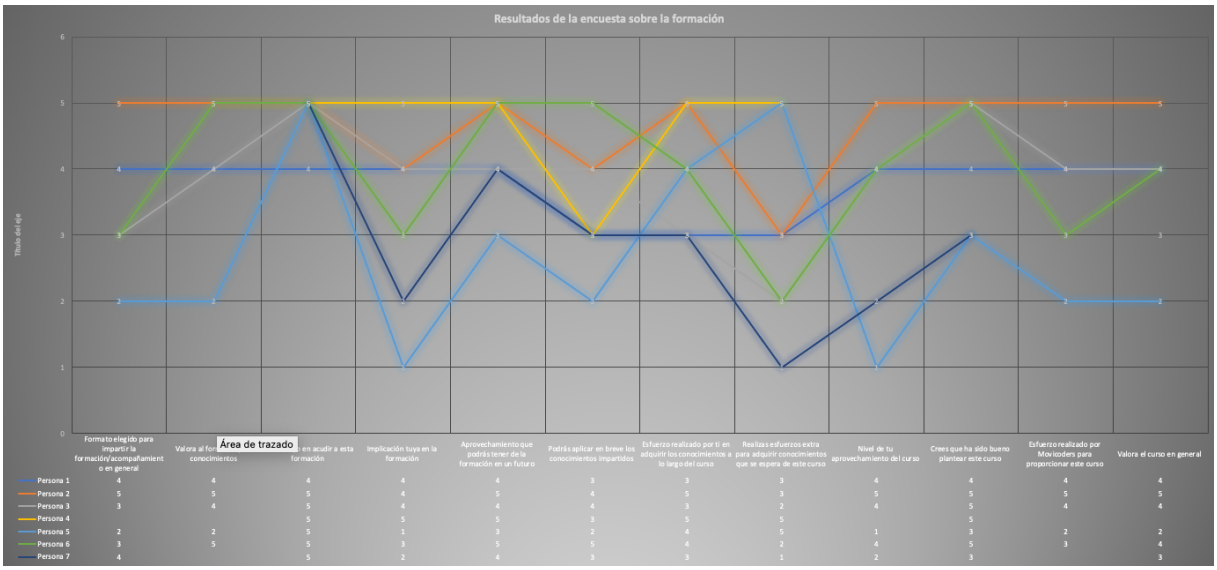


Figura 35. Resultados encuesta formación.

Si calculamos las medias de las respuestas para cada pregunta, los resultados obtenidos son:

Tabla 4. Resultados medios de la encuesta de formación por pregunta.

Pregunta	Valoración Media
Formato elegido para impartir la formación/acompañamiento en general	3,50
Valora al formador y sus conocimientos	4,00
Interés personal en acudir a esta formación	4,86
Implicación personal en la formación	3,29

Aprovechamiento que podrás tener de la formación en un futuro	4,29
Podrás aplicar en breve los conocimientos impartidos	3,43
Esfuerzo realizado por ti en adquirir los conocimientos a lo largo del curso	3,86
Esfuerzo extra realizado para adquirir conocimientos que se espera de este curso	3,00
Nivel de aprovechamiento del curso	3,33
Crees que ha sido bueno plantear este curso	4,29
Esfuerzo realizado por Movicoders para proporcionar este curso	3,60
Valora el curso en general	3,67

Resulta relevante que, las preguntas con una nota media más baja son:

- Esfuerzo extra realizado para adquirir conocimientos que se espera de este curso (3,00).
- Implicación personal en la formación (3,29).
- Nivel de aprovechamiento del curso (3,33).

Esto indica un nivel preocupante tanto de compromiso con la formación como con la mejora técnica y/o profesional personal, además de una baja motivación de la mayoría de los miembros del equipo.

Esto, siendo DevOps y, especialmente Agile, filosofías centradas en las personas, indica que existe un riesgo potencialmente alto de que esta formación, para este equipo, no sea una eficaz y pone en riesgo la implantación del proyecto en la empresa.

Por otro lado, si analizamos los datos desde el punto de vista de las personas y calculamos la nota media del cuestionario, obtenemos una nueva perspectiva que cualifica a los miembros del equipo en dos grupos diferenciados:

Tabla 5. Resultados encuesta formación por persona.

Persona	Valoración Media
Persona 1	3,75
Persona 2	4,67
Persona 3	3,83
Persona 4	4,71
Persona 5	2,67
Persona 6	4,00
Persona 7	3,00

En este caso, identificamos a determinado personal, que tiene un mayor grado de satisfacción con la formación (las personas 2, 4 y 6) y otros (las personas 5 y 7), que muestran cierta reticencia, desinterés o desaprovechamiento de la formación y probablemente deberían abandonar la misma, ya que no parece que puedan ser agentes del cambio.

Sobre los comentarios remitidos en la encuesta, se puede extraer lo siguiente:

- El formato de formaciones de 2 horas y su implementación en un proyecto en curso parece adecuado y dinámico para todos los miembros del equipo.
- El formador es adecuado y demuestra sus conocimientos y los sabe transmitir.
- El interés general es claro ya que les permite ampliar su conocimiento.
- Con el paso del tiempo, los miembros del equipo se implican más durante las sesiones.
- La tendencia general es que todos los participantes ven el objetivo a corto plazo y la aplicación de los conocimientos adquiridos.
- En general, entienden que la clave es la participación directa en un proyecto donde se aplique lo visto en la formación.

- El esfuerzo durante las sesiones es moderado, ya sea por la asistencia incompleta a las sesiones, porque el integrante no ve la aplicación directa o por falta de conocimientos.
- En general, el personal no realiza ningún esfuerzo extra fuera del horario laboral en profundizar o mejorar técnicamente.
- El aprovechamiento del curso va alineado con la participación en proyectos que utilicen lo visto en la formación.
- Todos entienden como positiva la propuesta formativa.
- Muchos no entienden la formación como un esfuerzo por parte de la empresa sino como una obligación.
- La valoración general es buena.

Desde una visión más global podemos decir que la formación es adecuada, pero las personas y sus circunstancias reducen el impacto positivo que esta podría tener en la empresa.

4.3.2. Desarrollo e implementación

De los objetivos de desarrollo propuestos, el estado final ha sido el siguiente:

Tabla 6. Resultados de los objetivos de desarrollo propuestos.

Objetivo	Resultado
Ampliar las **habilidades** técnicas de los desarrolladores.	No alcanzado
Aumentar la **calidad** del código entregable.	Alcanzado
Establecer un nivel técnico mínimo y **uniforme** entre los desarrolladores.	Alcanzado
Simplificar el desarrollo estableciendo una base sólida y propia.	Alcanzado
Incorporar TDD en **elementos clave** de nuestros desarrollos.	No alcanzado
Minimizar las **dependencias** de terceros.	No alcanzado

Por los resultados obtenidos en la evaluación de la formación, y principalmente en los miembros orientados a *frontend*, podemos decir que las habilidades técnicas de los participantes más experimentados no han sufrido variaciones relevantes, aunque sí se ha igualado el nivel base. Sin embargo, en *backend* si se han observado mejoras, especialmente en la calidad del código entregado al haber realizado un uso intensivo de tests (unitarios y de integración) y TDD en casos concretos. Aunque esta última técnica no se ha adoptado de

manera global. Además, redundando en esta mejora, se ha realizado la incorporación de pruebas *end-to-end* que garantizan un funcionamiento básico correcto.

Es cierto que no se están realizando métricas específicas y es difícil cuantificar esa mejora, sobre todo partiendo de un punto de partida inexistente, pero el *feedback* de los desarrolladores y del cliente es positivo en cuanto al bajo número de errores y retrabajos.

El desarrollo de librerías, plantillas y arquitecturas base ha simplificado y acelerado los desarrollos, especialmente en microservicios sencillos CRUD, donde se ha trivializado su implementación.

Las limitaciones temporales y de recursos han requerido postergar la eliminación de dependencias de terceros. Haciendo un balance de esfuerzo necesario y beneficios obtenidos, se ha desestimado invertir en ese aspecto.

4.3.3. Automatización

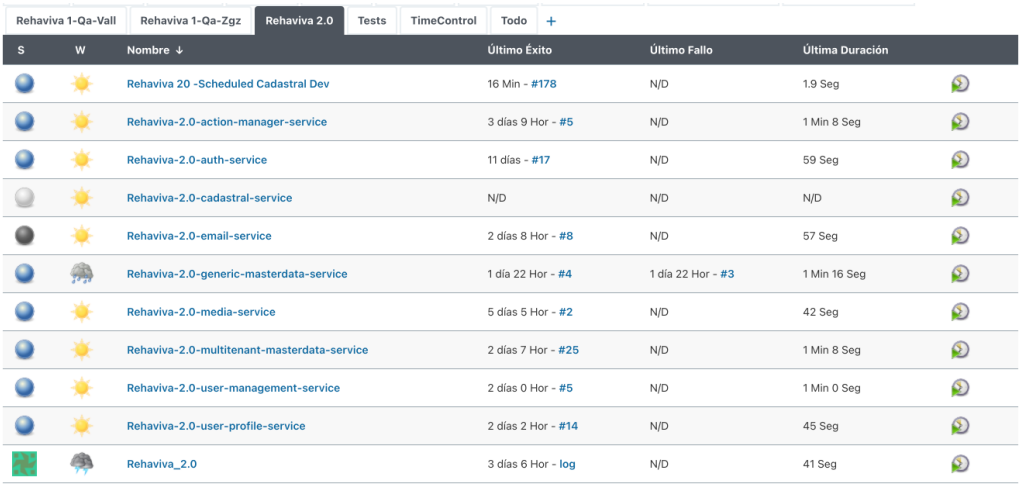
En cuanto a los objetivos de automatización, los resultados obtenidos son:

Tabla 7. Resultados de los objetivos de automatización propuestos.

Objetivo	Resultado
Simplificar el proceso de **construcción** del código.	Alcanzado
Implementar técnicas de ***testing*** automatizado a distintos niveles.	Alcanzado
Diseñar una estrategia de ***releases*** adecuada.	No alcanzado
Democratizar el **aprovisionamiento** de infraestructura.	No alcanzado
Automatizar despliegues de infraestructura en distintos entornos.	Alcanzado
Añadir **observabilidad** a las aplicaciones e infraestructuras.	Alcanzado
Establecer estrategias de ***backup***.	Alcanzado

La construcción de código se ha automatizado gracias a las herramientas Jenkins y Gradle, lo mismo que su distribución con Docker y ECR en AWS. Durante la ejecución del *pipeline* se ha introducido una etapa de ejecución de tests. Además, mediante la parametrización y con una estrategia basada en ramas, se ha conseguido desplegar en distintos entornos todos los componentes de la aplicación.

Como sucede en otras métricas, no existía un punto de partida que nos dé una referencia numérica de la mejora en conceptos como los periodos de entrega, pero mirando las ejecuciones satisfactorias de los *pipelines,* podemos observar que muchos de los servicios en desarrollo, tienen menos de 3 días desde el último despliegue en la siguiente imagen.



Rehaviva 1-Qa-Vall | Rehaviva 1-Qa-Zgz | Rehaviva 2.0 | Tests | TimeControl | Todo | +

S	W	Nombre ↓	Último Éxito	Último Fallo	Última Duración
		Rehaviva 20 -Scheduled Cadastral Dev	16 Min - #178	N/D	1.9 Seg
		Rehaviva-2.0-action-manager-service	3 días 9 Hor - #5	N/D	1 Min 8 Seg
		Rehaviva-2.0-auth-service	11 días - #17	N/D	59 Seg
		Rehaviva-2.0-cadastral-service	N/D	N/D	N/D
		Rehaviva-2.0-email-service	2 días 8 Hor - #8	N/D	57 Seg
		Rehaviva-2.0-generic-masterdata-service	1 día 22 Hor - #4	1 día 22 Hor - #3	1 Min 16 Seg
		Rehaviva-2.0-media-service	5 días 5 Hor - #2	N/D	42 Seg
		Rehaviva-2.0-multitenant-masterdata-service	2 días 7 Hor - #25	N/D	1 Min 8 Seg
		Rehaviva-2.0-user-management-service	2 días 0 Hor - #5	N/D	1 Min 0 Seg
		Rehaviva-2.0-user-profile-service	2 días 2 Hor - #14	N/D	45 Seg
		Rehaviva_2.0	3 días 6 Hor - log	N/D	41 Seg

Figura 36. Frecuencia de despliegues global de Rehaviva Crece.

Además, se ha observado que, desde la definición de un requisito hasta su entrega en producción, el tiempo se ha ido reduciendo con el tiempo hasta los cinco días actuales.

También es destacable ver el número de entregas que se ha hecho de un repositorio concreto:

Stage View

	Declarative: Checkout SCM	Clone repository	Compile gradle	Build, tag and push docker image to AWS	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 1s)	3s	544ms	41s	13s	1s	57ms
#106 Jul 08 11:49 — 1 commit	2s	494ms	34s	13s	1s	54ms
#105 Jul 08 08:51 — 1 commit	1s	551ms	34s	13s	1s	63ms
#104 Jul 08 08:25 — 5 commits	6s	671ms	55s	12s	1s	62ms
#103 Jul 07 14:18 — 1 commit	1s	461ms	43s	12s	1s	57ms
#102 Jul 07 13:15 — 1 commit	3s	546ms	36s	13s	1s	52ms

Figura 37. Entregas de un microservicio tipo.

En la anterior imagen se puede observar que la frecuencia de entrega es de varias veces al día en el entorno de desarrollo y eso se traduce en una entrega al día en producción.

Entre uno de los aspectos más relevantes, pero no alcanzados, se encuentra el diseño de una estrategia de *releases* correcta. Este objetivo ha pasado a ser la prioridad número uno del equipo dado el impacto que puede tener en nuestros proyectos.

Si bien hay más miembros del equipo con acceso al despliegue de infraestructura, no se puede decir que se haya democratizado. Se ha observado que no todo el personal tiene la capacidad y/o voluntad de adquirir ese tipo de responsabilidades.

Por el momento, la observabilidad de las aplicaciones e infraestructuras se lleva a cabo tal como se planificó para una primera etapa, mediante CloudWatch y OpenSearch. Si bien se identifica un margen de mejora y optimización en ambos casos.

Finalmente, se puede concluir que existe una estrategia de backups, tanto a nivel de datos, con copias regulares de la base de datos del entorno productivo gracias a RDS, como de los entregables gracias a ECR.

4.3.4. Objetivos socioculturales

Los objetivos socioculturales propuestos son los más complejos del total de los propuestos. Hacer que este tipo de cambios cale en un equipo requiere de un esfuerzo prolongado y continuo. Los resultados son los siguientes:

Tabla 8. Resultados de los objetivos socioculturales propuestos.

Objetivo	Resultado
Incorporar y fomentar la **cultura** DevOps en la empresa.	No alcanzado
Mejorar la **comunicación** entre los miembros de los equipos.	No alcanzado
Dar a **conocer** la arquitectura, tecnologías y herramientas necesarias.	Alcanzado
Diseñar un **plan de seguridad** frente a incidentes críticos no cubiertos por la plataforma.	No alcanzado

Incorporar la cultura DevOps en la empresa no ha sido posible aún. Mucho de este cambio se basa en los motores del cambio (personas), la voluntad de estas y el éxito del proyecto subyacente. Esto no es algo que sea sencillo, pero se han observado ciertos miembros con una mayor inclinación al cambio que serán los pilares sólidos sobre los que construir, y eso es tan importante como haberlo hecho.

Mejorar la comunicación, principalmente entre los equipos de desarrollo y operaciones, también está en proceso ya que todavía existe cierta reticencia en ambos equipos a esa unión. Se entiende que es un cambio difícil de asimilar y que las tareas del día a día no favorecen esa comunicación ya que aún sigue en la mentalidad de la empresa el antiguo paradigma del equipo de sistemas y el equipo de consultoría.

Gracias a la formación y a sesiones puntuales de transferencia de conocimiento, además de las sesiones de *Kick-off*, se ha dado a conocer la arquitectura, tecnologías y herramientas. Prueba de ello es la puesta en marcha del proyecto y las evidencias presentes en el proyecto donde se observa la metodología y herramientas ágiles aplicadas.

Finalmente, no podemos decir que haya un plan de seguridad de la aplicación, pero, durante el periodo de desarrollo de este proyecto, Movicoders S.L. ha conseguido la certificación ISO:27001, lo que garantiza, como mínimo, el compromiso de esta con la seguridad.

5. Conclusiones y trabajo futuro

A continuación, se exponen las conclusiones a este proyecto y su repercusión en la empresa tras la aceptación por parte de todos los integrantes del equipo de esta cultura y prácticas.

5.1. Conclusiones

Como se ha podido ver DevOps y las metodologías ágiles van de la mano, no en vano algunas de las técnicas propias de DevOps, como los tests automatizados y paradigmas de desarrollo como TDD han sido ideadas por algunos de los miembros del manifiesto ágil.

Implementar técnicas y herramientas DevOps tiene aún más sentido en proyectos y en productos, donde el equipo ve beneficiado su desarrollo personal y técnico, y sobre todo donde existen mayores posibilidades de éxito de la implantación.

Para el "meta-proyecto" propuesto, es fundamental el éxito de las pequeñas acciones realizadas en los proyectos guía. Y que el equipo involucrado en esas acciones sea capaz de transmitir ese éxito, ese foco en las personas, y esas dinámicas al resto de la organización.

Las metodologías ágiles son un vehículo dinámico, que debe evaluarse para cada proyecto y circunstancias en las que se apliquen. Deben adaptarse a cambios, necesidades y equipos, de forma natural. Si esto es así, la organización se verá beneficiada a distintos niveles.

Se ha hecho palpable la importancia de la selección de los miembros correctos para el equipo DevOps. Una de las principales preocupaciones que se desprenden del proyecto es la idoneidad del personal seleccionado dado el *feedback* obtenido. La experiencia o madurez, tanto técnica como (y principalmente) personal, es un factor determinante para acometer este cambio y hay que ser muy consciente del equipo con el que se trabaja en esos términos. Aun así, se entiende que queda trabajo por hacer en distintos aspectos, y que es algo corregible si se consigue mantener la confianza de la Dirección.

También se ha puesto de manifiesto la importancia de las métricas que soporten los resultados con algo más que apreciaciones personales de los miembros del equipo. Obviamente esta ha sido una etapa muy temprana de adopción de la cultura DevOps y debe entenderse como tal, pero es crítico empezar a obtener una perspectiva objetiva del avance del equipo y sus desarrollos.

Queda mucho margen de mejora y muchos objetivos por cumplir, pero a nivel técnico no se observa ningún riesgo que comprometa su alcance en medio plazo mientras haya un mínimo de componentes que crean en lo que hacen y que tengan empuje para llevarlo a cabo, será solo una cuestión de tiempo.

En líneas generales, Movicoders S.L. ha comenzado a experimentar el cambio y se ha visto reflejado en el día a día y en particular en el área de Nuevos Desarrollos. Esta cada vez ha ido cogiendo más peso hasta situarse como un referente tecnológico transversal.

Por parte de Dirección las conclusiones a este inicio son positivas y comparten las observaciones mencionadas anteriormente en cuanto a los resultados. Observan cierto crecimiento en miembros puntuales del área y en la empresa, donde estos avances y mejoras se ven como un pilar sobre el que apoyar con paso firme la decisión que se tomó al inicio del proyecto.

A nivel personal, este proyecto ha sido una experiencia exigente y algo agridulce, aunque con un balance general positivo. El hecho de apostar por personas y que el resultado no sea el esperado es complejo de gestionar, pero, cuando las personas responden y comparten el entusiasmo, el esfuerzo y la pasión por lo que hacen, todo lo que uno haya llevado a cabo se ve recompensado. En mi caso, para bien o para mal, eso es así… me esfuerzo al máximo, movido por la pasión por la tecnología, creo en lo que hago y sé que ese es el único camino que lleva al éxito.

5.2. Líneas de trabajo futuro

En la implantación de la cultura DevOps en la empresa Movicoders quedan pasos por dar como los siguientes:

- Fomentar la comunicación y cooperación entre desarrollo y operaciones hasta difuminar las barreras existentes.
- Motivación o reconfiguración del equipo que actuará como motor del cambio.
- Implantación de SonarQube para evaluar la calidad del código y detectar posibles puntos de fallo o errores.
- Mejora de los tiempos de ejecución de *pipelines frontend* para mejorar la experiencia de desarrollo del equipo.

- Diseñar una estrategia de *releases* correcta que facilite tanto la reutilización de los componentes como la identificación y trazabilidad de errores.
- Democratizar los despliegues de infraestructura para evitar cuellos de botella.
- Incorporar IaC con Terraform para evitar el *vendor-locking* existente actualmente con CloudFormation.

Todos estos apartados y algunos otros pueden mejorarse y ampliarse dentro de Movicoders, pero hay algo distintivo en cuanto a democratizar e introducir IaC en la empresa.

Respecto a estos dos últimos puntos, se han identificado algunos problemas que podrían derivar en una compleja incorporación a la empresa. Para los desarrolladores, adquirir responsabilidades de los despliegues de infraestructura les resulta una carga compleja y ajena. Para el personal de operaciones, supone un riesgo permitir accesos al despliegue de infraestructura a personal no familiarizado con sus procesos y conceptos propios. Además, observan el cambio como una intromisión que afecta a sus puestos de trabajo.

Por un lado, deberíamos hacer comprender el porqué de este cambio, que es una tarea más psicológica que técnica. Y por el otro, debemos simplificar dicha tarea y llevarla a un punto común, a un lenguaje común, que facilite la transición de las áreas hacia la cultura DevOps.

Durante el periodo de análisis de la incorporación de Terraform como IaC, tanto Operaciones como Desarrollo han encontrado un alto grado de dificultad. Al analizar las arquitecturas que se han propuesto para proyectos como Rehaviva Crece, todas tienen una línea común y siguen un esquema similar.

Haciendo una equivalencia con el mundo del desarrollo puro, el **dominio** de nuestras infraestructuras es muy **acotado y repetitivo**. Esas características, bajo un prisma DevOps, hacen nuestra IaC susceptible de ser automatizada. En programación, existe lo que se conoce como Model Driven Development (MDD), que consiste en desarrollar utilizando modelos acotados a un dominio y mucho más cercanos a él, de manera independiente de la plataforma y en un lenguaje próximo al del negocio. Para los despliegues de infraestructura, su equivalente sería Model Driven Infrastructure (MDI), y se ha planteado la posibilidad de proponer algo similar a la propuesta de Dougherty, White y Schmidt (2012) o Sandobalin, Insfran y Abrahão (2019), solo que, en este caso, el dominio serían despliegues de arquitecturas basadas en microservicios en AWS.

Para llevar a cabo la implementación de esta posible herramienta, se debe definir un Lenguaje Específico de Dominio (DSL), a través de la definición de sus abstracciones: meta-modelos y meta-metamodelos.

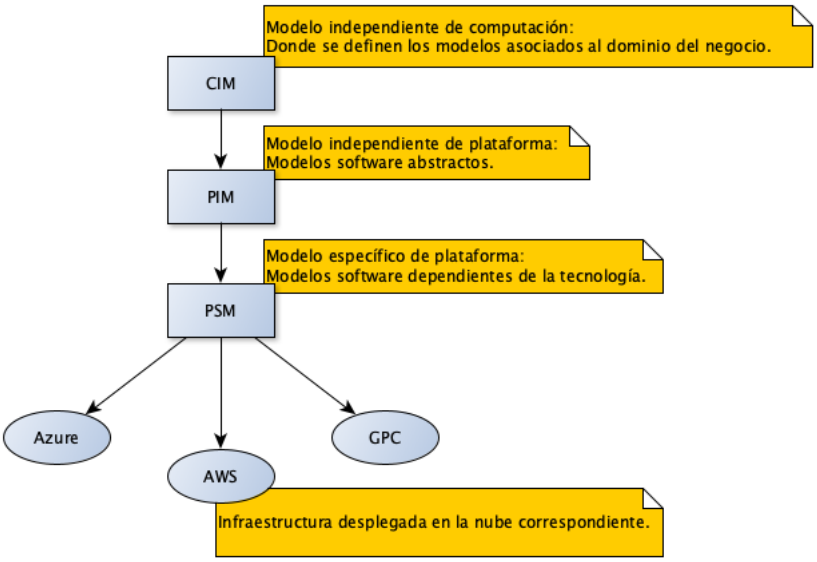


Figura 38. Diagrama de implementación de un sistema MDI.

Previamente ya había desarrollado una herramienta que, utilizando como base Eclipse, Ecore y EMF, permitía generar aplicaciones móviles para un dominio concreto (catálogos de venta de productos). En este caso, el proceso sería muy similar, pero con un dominio distinto, a la hora de la implementación del Modelo Específico de Plataforma (PSM) el objetivo sería generar, por ejemplo, llamadas al CLI de AWS. Este modelo, debería poder ser extensible a otras plataformas como GPC o Azure mediante la implementación del traductor a nivel de PSM de cada una de ellas.

Obviamente, esto inicia una línea de trabajo con mucho desarrollo por delante, que incluso se podría proponer como herramienta Open Source, con muchos puntos abiertos. Aunque bajo una perspectiva pragmática y para una casuística tan cerrada como la que Movicoders requiere, existiría la posibilidad de tener un MVP a corto/medio plazo.

Referencias bibliográficas

AEICE. (2022). AEICE y sus socios consiguen 1.200.000 euros de los fondos Next Generation EU para la mejora de la competitividad de las empresas en Castilla y León. https://web.archive.org/web/20220419114137/https://www.aeice.org/mincotur-extra-2021/

Beck, K. et al. (2001). The Agile Manifesto. Manifesto for Agile Software Development. Agile manifiesto. https://agilemanifesto.org/

Cubo, J. (Comp.). (2022). Marco ágil Scrum. Gestión de Proyectos. Universidad de la Rioja.

Corbin, Richard D. & Dunbar, Christopher B. & Zhu, Qiuming. (2007). A three-tier knowledge management scheme for software engineering support and innovation. J. Syst. Softw. 80(9), pp. 1494-1505.

Dougherty, B., White, J., & Schmidt, D. C. (2012). Model-driven auto-scaling of green cloud computing infrastructure. Future Generation Computer Systems, 28(2), 371-378.

Erich, Floris & Amrit, Chintan & Daneva, Maya. (2014). Report: DevOps Literature Review. 10.13140/2.1.5125.1201.

Feitelson, Dror & Frachtenberg, Eitan & Beck, Kent. (2013). Development and Deployment at Facebook. Internet Computing, IEEE. 17. 8-17. 10.1109/MIC.2013.25.

Hoare, Tony. (2009). Null References: The Billion Dollar Mistake. https://www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare/

Hosono, S. & Shimomura, Y. (2012). Application lifecycle kit for mass customization on PaaS platforms. In: Honolulu, HI, pp. 397–398.

ICCL. (2022). Rehaviva Crece. http://www.iccl.es/certificacion/rehaviva-crece

Latorre, Roberto. (2014). A successful application of a Test-Driven Development strategy in the industrial environment. Empirical Software Engineering. 19. 753-773. 10.1007/s10664-013-9281-9.

Mohamed, Samer. (2016). INNOVATIVE SOFTWARE DELIVERY FRAMEWORK TO WARDS SOFTWARE APPLICATIONS MODERNIZATION. International Journal of Research in Engineering & Technology. 4, pp. 77-98.

Sandobalin, J. & Insfran, E. & Abrahão, S. (2019). ARGON: A Model-Driven Infrastructure Provisioning Tool. ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C), 2019, pp. 738-742, doi: 10.1109/MODELS-C.2019.00114.

Schwaber, K. & Sutherland, J. (2020). La Guía Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego. scrumguides.org. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

Stuckenberg, Sebastian & Fielt, Erwin & Loser, Timm. (2011). The Impact Of Software-As-A-Service On Business Models Of Leading Software Vendors: Experiences From Three Exploratory Case Studies. PACIS 2011 - 15th Pacific Asia Conference on Information Systems: Quality Research in Pacific, pp. 184.

Anexo A. Resultados encuesta de formación

Pregunta	Persona 1	
Formato elegido para impartir la formación/acompañamiento en general	4	El formato es bastante dinámico y ayuda a la comunicación entre compañeros.
Formador y sus conocimientos	4	El formador es un buen comunicador y facilita el entendimiento de varios conceptos que, a priori, pueden llegar a ser confusos. Sus amplios conocimientos ayudan a ver distintas formas de programar.
Interés personal en acudir a esta formación	4	Bastante, la nueva arquitectura planteada es bastante interesante y más dinámica que las anteriores, aunque es verdad que no hemos podido avanzar mucho ni verla en profundidad por falta de recursos.
Implicación personal en la formación	4	Al principio me podía la timidez y no participaba mucho, aunque ahora intento implicarme más.
Aprovechamiento de la formación en un futuro	4	Las distintas formas que nos ha enseñado Damián se pueden aplicar en bastantes campos y se puede aprovechar bien.
Aplicabilidad a corto plazo de los conocimientos impartidos	3	Hasta ahora no hemos podido aplicar muchos de los conceptos que Damián nos ha impartido, aunque es cierto que ahora que empezamos a tener los recursos necesarios en el proyecto para progresar sí que se podrán aplicar más.
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	3	Muchos de los conocimientos se pueden aplicar en cosas muy específicas, aunque siempre es bueno tenerlo presente, no muchas veces se pueden aplicar.
Esfuerzo personal extra para profundizar en la formación	3	Intento leer todos los documentos que suele enviar y ampliar un poco los conceptos que plantea, aunque hay veces que no hay tiempo para hacerlo.
Nivel de aprovechamiento del curso	4	
Idoneidad del planteamiento del curso	4	
Esfuerzo realizado por la empresa para proporcionar este curso	4	
Valoración del curso en general	4	

Pregunta	Persona 2	
Formato elegido para impartir la formación/acompañamiento en general	5	
Formador y sus conocimientos	5	
Interés personal en acudir a esta formación	5	
Implicación personal en la formación	4	

Aprovechamiento de la formación en un futuro	5	
Aplicabilidad a corto plazo de los conocimientos impartidos	4	
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	5	
Esfuerzo personal extra para profundizar en la formación	3	Intento dedicarle algún tiempo por las tardes o en el fin de semana, pero no siempre puedo.
Nivel de aprovechamiento del curso	5	
Idoneidad del planteamiento del curso	5	Creo que hemos sacado muy buenas prácticas de este curso, como los tests. También creo que es importante para Movicoders haber creado una arquitectura propia, ya que acelera el desarrollo y hace que dependamos de menos librerías externas.
Esfuerzo realizado por la empresa para proporcionar este curso	5	
Valoración del curso en general	5	

Pregunta	Persona 3	
Formato elegido para impartir la formación/acompañamiento en general	3	
Formador y sus conocimientos	4	El formador es un bueno, tiene paciencia con nosotros y nos anima a participar. Tiene su forma de hacer las cosas y su mentalidad a la hora de programar, la cual intenta inculcarnos. La pregunta es si la empresa tiene interés en tomar esa filosofía al completo.
Interés personal en acudir a esta formación	5	Alto. Cuando empezamos el curso no tenía ni idea, ya no de la arquitectura propuesta, si no de ninguna en absoluto más allá de la aplicada en podoactiva. Esta era una oportunidad de que alguien me explicar cómo aplicar y comprender una arquitectura de proyecto de frontend.
Implicación personal en la formación	4	He de reconocer que al principio me encontraba más reticente a participar e implicarme. Pero actualmente procuro participar más e involucrarme lo máximo posible. Todo lo que el hecho de dedicar el resto de mi tiempo al Portal y no a Rehaviva me permite.
Aprovechamiento de la formación en un futuro	4	Si se adopta las metodologías y arquitectura impartidas en el mismo en los proyectos de forma concienzuda, si por supuesto
Aplicabilidad a corto plazo de los conocimientos impartidos	4	Lo intento, aunque en el Portal me veo limitado por la aplicación y de una metodología de programación y una estructura predefinida que nos son las promulgadas en el curso. Cuando entre a Rehaviva podré aprovecharlo mucho más (en caso de que se aplique claro).

Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	3	He intentado dar lo mejor de mí en todo momento. Mientras que si es cierto que algunas sesiones me he perdido y al perder el hilo completamente no sabía ni siquiera qué preguntar. Aunque esto último me ocurre cada vez con menos frecuencia.
Esfuerzo personal extra para profundizar en la formación	2	Suelo revisar el proyecto de Rehaviva y los avances para intentar aplicar todo lo posible en mi actual proyecto (para lo que tengo bastantes limitaciones) pero me ayuda a repasar lo hecho hasta ahora.
Nivel de aprovechamiento del curso	4	Hasta ahora he podido aprovechar poco por lo explicado en la pregunta anterior, en el Portal me veo limitado a aplicarlo por la arquitectura que ya se había elegido para dicho proyecto. Todavía no me encuentro trabajando en Rehaviva. Por otra parte, si es cierto que la formación me está ayudando a mentalizarme en cuanto a una filosofía de programación que me está siendo útil. Cuando entre a Rehaviva podré aplicar y aprovechar este conocimiento más exhaustivamente.
Idoneidad del planteamiento del curso	5	Creo que sí, este cambio de arquitectura les sentará bien a los proyectos que lo apliquen. Tanto en funcionamiento de estos como en facilidad a la hora de trabajar en ellos.
Esfuerzo realizado por la empresa para proporcionar este curso	4	No puedo valorar este aspecto más allá del esfuerzo que supone contratar a una persona externa para impartirlo.
Valoración del curso en general	4	

Pregunta	Persona 4	
Formato elegido para impartir la formación/acompañamiento en general		
Formador y sus conocimientos		
Interés personal en acudir a esta formación	5	Si bien es cierto que no he podido asistir al curso desde su inicio, considero que este tipo de formaciones son necesarias y pueden ser de mucha utilidad a programadores tanto juniors como seniors. Personalmente, voy a aprender de Damián todo lo posible.
Implicación personal en la formación	5	Tengo mucho interés en poder participar en las reuniones y aportar conocimientos, así como aprender de mis compañeros.
Aprovechamiento de la formación en un futuro	5	Al estar realizando el curso sobre una biblioteca tan actual y por lo que parece con muchos años de vida, me servirá todo lo aprendido para aplicarlo en un futuro.
Aplicabilidad a corto plazo de los conocimientos impartidos	3	Espero que cuanto antes.
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	5	

Esfuerzo personal extra para profundizar en la formación	5	Estoy realizando varios cursos fuera del horario de la empresa sobre arquitectura de software para aprender más.
Nivel de aprovechamiento del curso		
Idoneidad del planteamiento del curso	5	Ha sido muy bueno plantearlo; espero que se sepa aprovechar. Y además que plantee una base sólida y bien definida sobre la que se puedan soportar los proyectos venideros.
Esfuerzo realizado por la empresa para proporcionar este curso		
Valoración del curso en general		

Pregunta	Persona 5	
Formato elegido para impartir la formación/acompañamiento en general	2	
Formador y sus conocimientos	2	
Interés personal en acudir a esta formación	5	
Implicación personal en la formación	1	
Aprovechamiento de la formación en un futuro	3	
Aplicabilidad a corto plazo de los conocimientos impartidos	2	
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	4	
Esfuerzo personal extra para profundizar en la formación	5	
Nivel de aprovechamiento del curso	1	
Idoneidad del planteamiento del curso	3	
Esfuerzo realizado por la empresa para proporcionar este curso	2	
Valoración del curso en general	2	

Pregunta	Persona 6	
Formato elegido para impartir la formación/acompañamiento en general	3	
Formador y sus conocimientos	5	
Interés personal en acudir a esta formación	5	
Implicación personal en la formación	3	
Aprovechamiento de la formación en un futuro	5	
Aplicabilidad a corto plazo de los conocimientos impartidos	5	
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	4	Cuando estábamos hablando de la arquitectura y forma de hacer el proyecto estaba muy interesado e implicado pero ha ido cayendo el interés al alejarnos y dedicarle más tiempo del necesario a cosas que no tiene sentido dedicarle mucho esfuerzo más allá de ver un par de ejemplos y saber que existe. Ya que como muchas veces

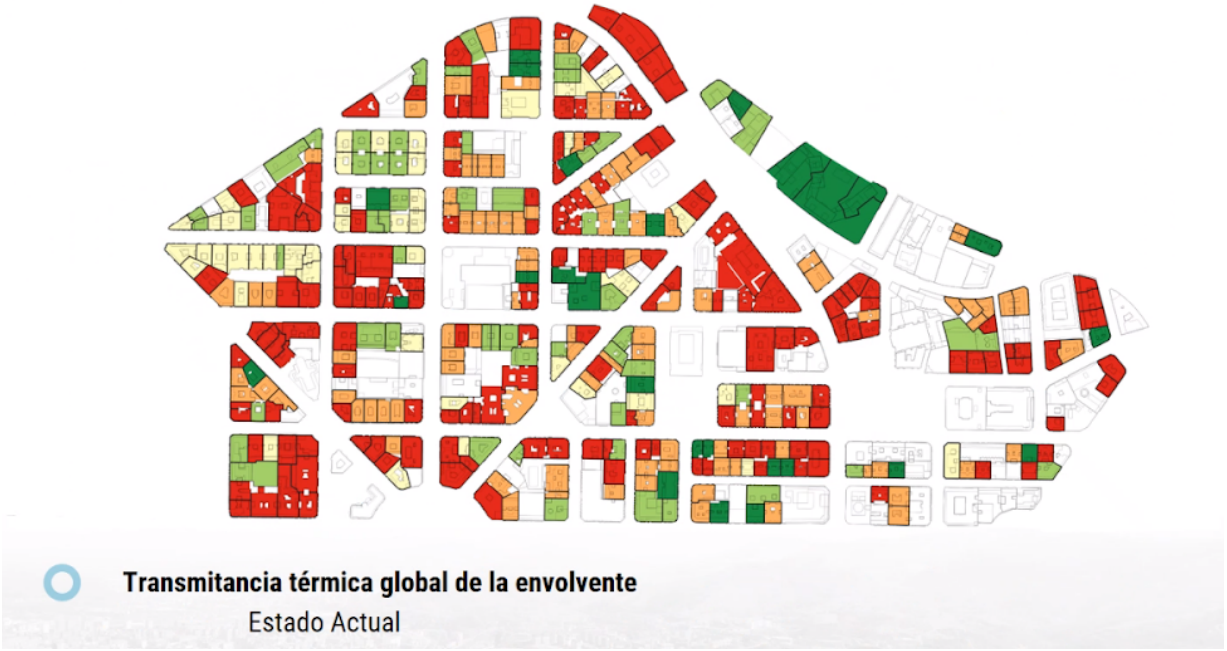
		ha mencionado a lo largo de las sesiones, hay que practicar y utilizar lo que aprendemos en estas sesiones durante mucho tiempo para quedarse de verdad con los conceptos y, para usar algo reciente de esta mañana. Hacer 2 días TDD y o katas (que son ejercicios de programación que deberíamos hacer fuera del trabajo cosa que muchos de nosotros no vamos a hacer porque aunque a algunos nos guste programar no es necesariamente nuestra pasión) si no vamos a hacerlo a la hora de trabajar (que ese es otro tema, si utilizamos absolutamente todo lo que aprendemos de vez o solo cuando es necesario) no es muy eficaz porque con practicar 4 horas algo no vas a coger ni costumbre ni aprender más que los primeros 20 minutos. Esto se puede aplicar a cosas parecidas.
Esfuerzo personal extra para profundizar en la formación	2	Las primeras semanas intenté participar en la arquitectura y añadir ahí cosas que nos fuera diciendo el formador, pero, como cuando pedía revisar esas cosas para que viéramos todos cómo debemos actuar a la hora de añadir cosas o colaborar con la arquitectura no se me hacía caso y no revisamos esas cosas, no estoy haciendo nada nuevo a parte de implementar lo que aprendo en Rehaviva.
Nivel de aprovechamiento del curso	4	
Idoneidad del planteamiento del curso	5	
Esfuerzo realizado por la empresa para proporcionar este curso	3	Es una inversión de horas de los que están involucrados. En cuanto a lo que se invierte en el formador sin estar nosotros debería ser el mismo al ser una consultoría para realizar una arquitectura que se realizaría de cualquier manera sin o con nosotros.
Valoración del curso en general	4	

Pregunta	Persona 7	
Formato elegido para impartir la formación/acompañamiento en general	4	Un formato de aprender haciendo de lo más productivo.
Formador y sus conocimientos		Es una persona con un amplio conocimiento de la materia.
Interés personal en acudir a esta formación	5	Me interesa mejorar como programador React para poder enfrentarme a los retos laborales que la empresa me plantea.
Implicación personal en la formación	2	Poca, tan solo puedo asistir a una de las dos horas de la formación, ya que mi tiempo está vendido a Osapiens, tampoco tengo tiempo para hacer las actividades adicionaes que plantea.

Aprovechamiento de la formación en un futuro	4	Plantea cosas interesantes sobre desarrollo con tests.
Aplicabilidad a corto plazo de los conocimientos impartidos	3	El modelo de arquitectura que se emplea en los proyectos en los que trabajo con Osapiens no es el que imparte la formación.
Esfuerzo personal realizado en adquirir los conocimientos a lo largo del curso	3	Esfuerzo acorde a mi asistencia.
Esfuerzo personal extra para profundizar en la formación	1	No invierto tiempo fuera de la formación, ni en el resto de mi jornada laboral ni fuera de ella.
Nivel de aprovechamiento del curso	2	No puedo estar realmente enfocado en este curso por mis compromisos laborales.
Idoneidad del planteamiento del curso	3	No puedo responder en este momento.
Esfuerzo realizado por la empresa para proporcionar este curso		No me siento capaz de valorar esto.
Valoración del curso en general	3	No lo estoy pudiendo aprovechar tanto como me gustaría.

Anexo B. Ejemplos de expectativas de UI para Rehaviva IA

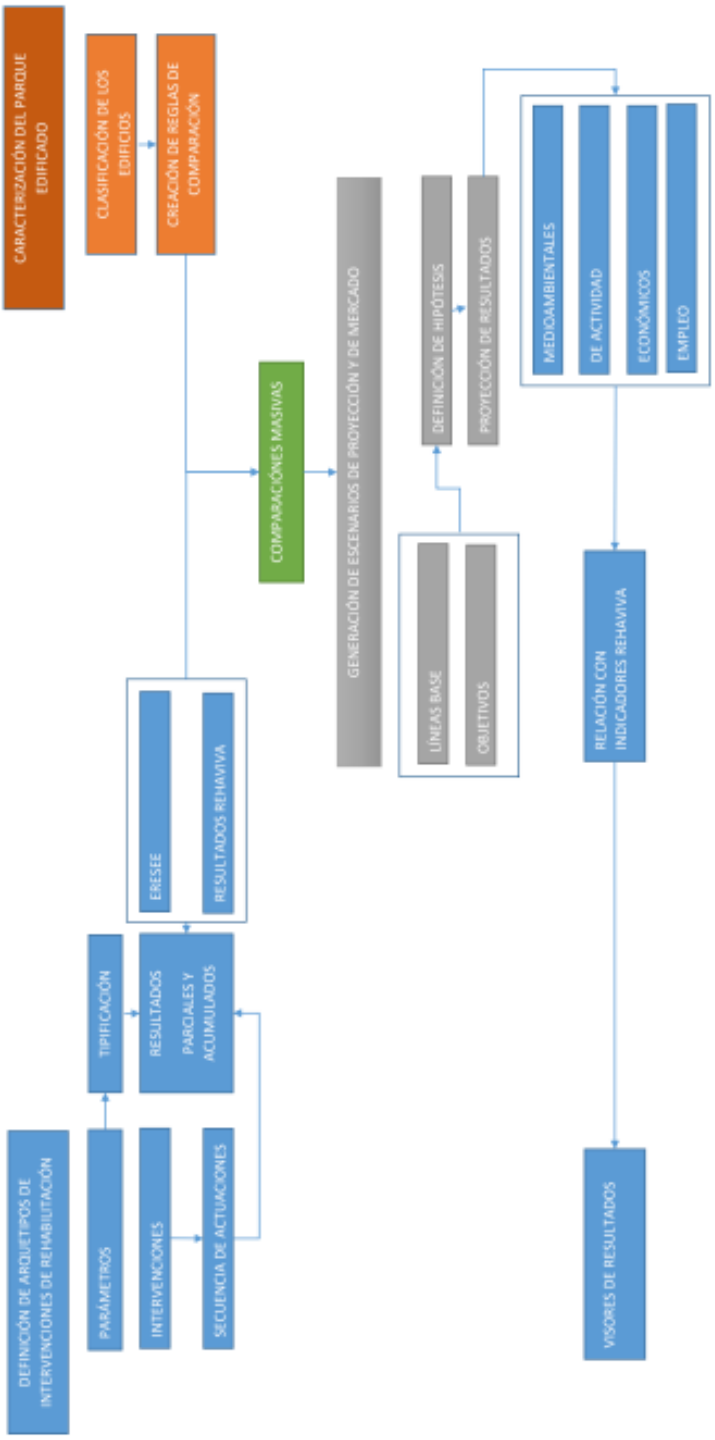
Uso principal del inmueble



Transmitancia térmica global de la envolvente

Estado Actual

Anexo C. Flujo base para Rehaviva IA



Anexo D. Ejemplo de definición de especificaciones

APLICACIÓN REHAVIVA – CRECE (Escala municipal)		
BLOQUES FUNCIONALES	**CONCEPTO**	**DESCRIPCIÓN RESUMIDA**
BLOQUE FUNCIONAL 1	**GESTIÓN DE ACTUACIONES DE REHABILITACIÓN REALIZADAS EN LA CIUDAD**	Basado en la estructura base de Rehaviva a la que se le incluyen correcciones o mejoras detectadas en las pruebas piloto
BLOQUE FUNCIONAL 2	**CÁLCULO DE INDICADORES DE REHABILITACIÓN SOBRE ACTUACIONES REALES EN LA CIUDAD**	Basado en la estructura base de Rehaviva a la que se le incluyen correcciones o mejoras detectadas en las pruebas piloto
BLOQUE FUNCIONAL 3	**CARACTERIZACIÓN DEL PARQUE EDIFICADO**	Basado en: - Los datos disponibles de las actuaciones de rehabilitación realizadas gestionadas a través del Bloque Funcional 1 de REHAVIVA - La clasificación tipológica de los edificios y la asignación de características geométricas y constructivas por comparación con arquetipos predefinidos - La estimación de consumos y emisiones por asignación de valores extraídos de las bases de datos de certificaciones energéticas registradas para edificios asimilables
BLOQUE FUNCIONAL 4	**GENERACIÓN DE ESCENARIOS DE PROYECCIÓN DE AHORRO ENERGÉTICO Y REDUCCIÓN DE EMISIONES**	Basados en la caracterización del parque edificado realizado en el Bloque Funcional 3. Permiten la definición de líneas base y diferentes hipótesis de trabajo. Devuelven informes de resultados y mapas representativos

BLOQUE FUNCIONAL 5	**GENERACIÓN DE ESCENARIOS DE PROYECCIÓN DE MERCADO Y POTENCIAL DE ACTIVIDAD**	Basados en la caracterización del parque edificado realizado en el Bloque Funcional 3. Permiten la definición de líneas base y diferentes hipótesis de trabajo. Devuelven informes de resultados y mapas representativos

Anexo E. Generación de escenarios de proyección