1. **TÍTULO DEL RETO**

Aplicación de Ayudas Públicas con IA Generativa (Bono alquiler)

Uso de la Inteligencia Artificial para la construcción de nueva aplicación de un nuevo servicio que presta una consejería, con enfoque fast-track.

1. **DESCRIPCIÓN DEL CASO DE USO**

* CONTEXTO: Se trata de un caso hipotético en el que una consejería va a poner en marcha una nueva ayuda pública y necesita dotarse de una aplicación que facilite su gestión.
* NECESIDAD: Construir un sistema que contemple dos aplicaciones:
  + **Aplicación del Ciudadano**, para su interacción con el servicio de ayudas, a través de internet.
  + **Aplicación del Gestor**, para la gestión del servicio por parte de la Administración.

1. **PROPÓSITO DEL HACKATHON**

* A partir del BOCM que describe las bases reguladoras del servicio, se deben construir las aplicaciones que permitan su gestión de manera end-to-end, es decir desde la solicitud por el ciudadano, hasta la tramitación completa por parte del gestor de la consejería.
* El reto consiste en conseguir el mayor número de logros posibles, descritos más adelante, coincidiendo con los hitos principales del ciclo de vida de desarrollo: (1) Definición de requisitos, (2) definición de la arquitectura y diseño software y (3) desarrollo.
* Para llevar a cabo las actividades, los participantes pueden apoyarse en las herramientas de IA que deseen, supervisando y adaptando el resultado cuando sea necesario.
* Por otro lado, dada la limitación de tiempo existente para desarrollar el reto, se recomienda aplicar un enfoque de trabajo tipo fast-track, lo cual implica llevar a cabo un análisis previo de las tareas y elaborar un plan de trabajo que permita paralelizar lo máximo posible las actividades, así como un reparto de roles entre los miembros del equipo.

1. **LOGROS**

LOGRO 1: Definición de requisitos, obtenido mediante ingeniería de requisitos.

LOGRO 2: Definición de la arquitectura y diseño software.

LOGRO 3: Desarrollo de los componentes y pruebas unitarias.

1. **TECNOLOGÍAS A UTILIZAR Y HERRAMIENTAS RECOMENDADAS**

Se especifica en cada uno de los LOGROS.

**LOGRO 1: Definición de requisitos, obtenido mediante ingeniería de requisitos**

1. **PROPÓSITO**

Se trata de poner en práctica las habilidades de los participantes en relación con los requisitos de software: obtención, análisis y documentación de mismos, del caso hipotético que será descrito con detalle al inicio del hackathon.

El responsable del servicio de ayudas de vivienda nos ha convocado para trasladarnos la necesidad de implementar un caso de uso que expresa en los siguientes términos:

“Necesitamos dos aplicaciones que nos permitan gestionar la ayuda *bono alquiler* destinada a jóvenes:

* + APLICACIÓN PARA LA SOLICITUD: aplicación que permita a la ciudadanía solicitar por internet la ayuda, de forma que se puedan informar de los requisitos mínimos y puedan llevar a cabo la solicitud introduciendo los datos y documentación necesaria. El detalle de requisitos e información a aportar en el proceso de solicitud se encuentra en el [documento de bases reguladoras de la ayuda](https://www.comunidad.madrid/sites/default/files/doc/vivienda/bono_joven_acuerdo_del_consejo_de_gobierno_bono_alquiler_joven.pdf). Cada solicitud dará lugar a un expediente, cuya tramitación se llevará a cabo desde la otra aplicación a construir.
  + APLICACIÓN PARA LA TRAMITACIÓN DE EXPEDIENTES: aplicación que permita a los gestores del servicio de ayudas proceder con la tramitación de cada expediente, desde un estado inicial “Pendiente de iniciar”, a través del siguiente flujo:
  + Elección del expediente a tramitar, pasando a estado “En revisión formal”.
  + Comprobación de que el solicitante ha aportado los documentos requeridos, pasando a estado “En revisión de requisitos” si la verificación tuvo éxito, o bien, a estado “En espera de subsanación” si alguna documentación es incorrecta o faltante, enviando un correo electrónico al solicitante para que lo subsane por ese medio.
  + Comprobar que, según documentación, el solicitante cumple los requisitos para percibir la ayuda. En caso afirmativo, se le comunica por correo electrónico y el expediente pasa a estado “Finalizado, favorable”. En caso contrario se le comunica por correo electrónico que no cumple los requisitos y por tanto se deniega, pasando el expediente a estado “Finalizado, desestimado”.

1. **ENTREGABLES**

* Documento pdf que recoja la **especificación de requisitos**, listo para ser entregado al responsable del servicio de ayudas. Este debe ser lo suficientemente claro y completo como para que pueda entenderlo y validarlo antes de iniciar la fase de implementación.

El documento deberá subirse a git, ubicándose en la carpeta:

reto1\_aplicacion/<nombre\_equipo>/logro1

* El README.md del logro1 deberá contener una explicación del procedimiento seguido y el apoyo obtenido usando herramientas de IA.

1. **INFORMACIÓN DE PARTIDA**

Enlace al [documento de acuerdo de gobierno](https://www.comunidad.madrid/sites/default/files/doc/vivienda/bono_joven_acuerdo_del_consejo_de_gobierno_bono_alquiler_joven.pdf) que incluye las bases reguladoras del programa de ayuda al alquiler de vivienda y de ayuda a las personas jóvenes y para combatir el reto demográfico.

1. **HERRAMIENTAS RECOMENDADAS**

**Se puede utilizar cualquier herramienta que suponga un acelerador.** A continuación, se exponen algunas herramientas recomendadas, basadas en IA:

* Chatbot de IA Generativa como ChatGPT o Gemini
* Herramientas de diagramación a través de script como, por ejemplo:
* [PlantUML](https://www.plantuml.com/) (formato puml)
* [Mermaid Live Editor](https://mermaid.live/edit) (formato mermaid)
* Herramienta que permita prototipado de pantallas como, por ejemplo:
* [Stitch](https://stitch.withgoogle.com/) (proporciona un guiado paso a paso)
* [Figma](https://www.figma.com)

1. **CRITERIOS DE EVALUACIÓN DE LOS ENTREGABLES** 
   1. **Alcance funcional**: El documento recoge todas las necesidades expresadas con el suficiente detalle como para ser validado por el responsable del servicio.
   2. **Estructura**: Dispone de una adecuada estructuración de las funcionalidades.
   3. **Comprensión**:lenguaje claro, términos sin ambigüedades, uso de gráficos explicativos, pantallas de ejemplo, etc.
   4. **Uso adecuado de la IA**

**LOGRO 2: Definición de la arquitectura y diseño software.**

1. **PROPÓSITO**

Una vez los requisitos han sido validados por el responsable del Servicio de Ayudas, es necesario llevar a cabo la definición de la arquitectura y el diseño del sistema atendiendo a los siguientes principios:

* Arquitectura según el patrón n-capas (Layered Architecture / N-Tier)
* Separación de la capa de presentación o frontend, en tecnología web, de la lógica de negocio o backend, basada en Springboot.
* Diseño del backend estructurándolo en servicios apificados y cumpliendo buenas prácticas de diseño de APIs, con el objetivo de que sean developer-friendly.
* Utilización del patrón BFF en la parte backend para separar la lógica dedicada a cada canal (servicio que atiende cada aplicación) de las funcionalidades comunes de negocio (servicios de negocio)
* Modelización de los datos para una base de datos relacional.

1. **ENTREGABLES**

* Documento de **arquitectura y diseño del software**, listo para ser entregado al responsable de Arquitectura en Madrid Digital para su VºBº, de modo que pueda dar comienzo la siguiente fase desarrollo.

El documento deberá subirse a git, ubicándose en la carpeta:

reto1\_aplicacion/<nombre\_equipo>/logro2

* El README.md del logro2 deberá contener una explicación del procedimiento seguido y el apoyo obtenido usando herramientas de IA.

1. **INFORMACIÓN DE PARTIDA**

Documento de análisis de requisitos resultante del logro anterior.

1. **HERRAMIENTAS RECOMENDADAS**

**Se puede utilizar cualquier herramienta que suponga un acelerador.** A continuación, se exponen algunas herramientas recomendadas, basadas en IA:

* Chatbot de IA Generativa como ChatGPT o Gemini
* Herramientas de diagramación a través de script como, por ejemplo:
* [PlantUML](https://www.plantuml.com/) (formato puml)
* [Mermaid Live Editor](https://mermaid.live/edit) (formato mermaid)

1. **CRITERIOS DE EVALUACIÓN DEL ENTREGABLE**
2. **Alcance**: El documento recoge la definición de la arquitectura y el diseño software, cumpliendo con los principios expresados, con suficiente detalle técnico como para ser validado por el responsable del Área de Arquitectura de Madrid Digital.
3. **Estructura**: Dispone de una adecuada estructuración de los contenidos.
4. **Comprensión**: Lenguaje claro, términos sin ambigüedades, uso de gráficos y diagramas explicativos.
5. **Uso adecuado de la IA**

**LOGRO 3: Desarrollo de los componentes y pruebas unitarias.**

1. **PROPÓSITO**

Una vez la arquitectura y el diseño del software ha sido validado por el responsable de Arquitectura de Madrid Digital, es necesario construir los componentes del sistema que se encuentran marcados en color verde en la siguiente lista:

|  |  |
| --- | --- |
| [Esta foto](https://iktnekane.blogspot.com/2018/04/ikt-en-abantailak-eta-desabantailak.html) de Autor desconocido está bajo licencia [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/) | ~~Frontal web de las dos aplicaciones, utilizando Angular o React~~ |
| [Esta foto](https://iktnekane.blogspot.com/2018/04/ikt-en-abantailak-eta-desabantailak.html) de Autor desconocido está bajo licencia [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/) | ~~Swagger de las APIs de los servicios BFF, empleando el estándar OpenAPI 3.0~~ |
| [Esta foto](https://iktnekane.blogspot.com/2018/04/ikt-en-abantailak-eta-desabantailak.html) de Autor desconocido está bajo licencia [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/) | ~~Backend RESTful para los servicios BFF, mediante Java y Springboot~~ |
|  | Especificación de las APIs de los servicios de negocio, usando el estándar OpenAPI 3.0 |
|  | Backend RESTful de los servicios de negocio, mediante Java y Springboot |
|  | Persistencia de datos en una base de datos relacional con el modelo de datos físico y datos de test |

El código fuente de los servicios backend, una vez desarrollado, debe cumplir lo siguiente:

* Debe contener un alto grado de cobertura de **pruebas unitarias**, implementadas con Junit.
* Debe haber pasado unos **test funcionales** completos, realizados con Postman.
* Se debe asegurar una buena **calidad del código**, demostrable a través del análisis de SonarQube.

1. **ENTREGABLES**

* Artefactos construidos y subidos a git, ubicándose se la siguiente forma en el repositorio:

|  |  |
| --- | --- |
| **ARTEFACTO** | **UBICACIÓN DE ENTREGA EN GIT** |
| Ficheros yaml de especificación de las APIs en Open API 3.0 | reto1\_aplicacion/<nombre\_equipo>/logro3/API |
| Carpetas de los proyectos Java SpringBoot de los servicios REST | reto1\_aplicacion/<nombre\_equipo>/logro3/backend |
| Fichero sql de creación del modelo de datos, y fichero sql de carga inicial de datos | reto1\_aplicacion/<nombre\_equipo>/logro3/persistencia |

* Evidencia de ejecución de los test funcionales de los servicios de negocio, realizados con Postman. Se deben subir a la siguiente ubicación de git los ficheros json de resultados, que se exportan desde Postman una vez ejecutados los test:

reto1\_aplicacion/<nombre\_equipo>/logro3/tests

* Se debe dejar una evidencia de ejecución del chequeo de calidad de los proyectos java en un SonarQube Server compartido durante el hackathon. La ejecución del análisis de código deberá vincularse al proyecto SonarQube identificado con el nombre del equipo de trabajo.
* El README.md del logro3 deberá contener una explicación del procedimiento seguido y el apoyo obtenido usando herramientas de IA.

1. **INFORMACIÓN DE PARTIDA**

Documento de arquitectura y diseño software generado en el logro anterior.

1. **TECNOLOGÍAS A UTILIZAR Y HERRAMIENTAS RECOMENDADAS**

**Frameworks y herramientas que OBLIGATORIAMENTE se deben emplear:**

* Java **Spring Boot 3.x** (implica tener instalado y configurado **JDK 17** o superior)
* Testing:
  + Pruebas unitarias: framework **Junit 5**
  + Pruebas funcionales: herramienta **Postman**

**IDE de desarrollo a utilizar:**

Se puede usar el IDE para desarrollo de Java y la herramienta de empaquetado Java preferidos, siempre y cuando puedan integrase con el análisis de código en SonarQube.

**A modo de recomendación, relacionamos un conjunto de herramientas que haría posible afrontar el logro:**

* **IDE VSCode**, con las extensiones necesarias para desarrollar con SpringBoot y Junit, además de integrar el control de calidad:
  + Java Extension Pack
  + Spring Boot Extension Pack
  + SonarQube (antiguo SonarLint)
  + Maven for Java
  + Debugger for Java
  + Test Runner for Java
* **Maven,** como herramienta de empaquetado Java.
* **PostgreSQL o SQLite**, como base de datos.
* Asistentes de Inteligencia artificial:
* Chatbot de IA Generativa como **ChatGPT** o **Gemini**
* **Codeium/Windsurf** o **GitHub Copilot**, como asistentes de desarrollo de IA, integrados en el IDE

1. **CRITERIOS DE EVALUACIÓN DEL ENTREGABLE**
2. **Alcance**: los componentes software que forman parte de la arquitectura han sido construidos, se encuentran adecuadamente documentados y el resultado de los test funcionales es conforme.
3. **Calidad**: tienen un alto grado de cobertura de test unitarios, por otro lado, el grado de calidad y deuda técnica reportada en el informe de calidad debería ser satisfactorio.
4. **Uso adecuado de la IA**