OFERTA PARA LA REALIZACIÓN DE TFM/TFG EN R

Condiciones

- Prácticas en empresas de 10 meses de duración
- Jornada: Tiempo parcial 15/20 horas semanales
- Remuneración: Dependiendo del perfil del candidato.
- Lugar de trabajo: Sedes de R en Vigo o Coruña. Posibilidad de teletrabajo.

Descripción: El estudiante se integrará en uno de los proyectos que se indican a continuación para realizar el TFM/TFG.

Contacto: Ana Fernández Vilas (avilas@det.uvigo.es UVIGO) y Carlos Dafonte (dafonte@udc.es UDC)

MASM-CTDR-1: Microservicios para distintas arquitecturas orientadas a eventos. El objetivo de este proyecto es el desarrollo de microservicios en arquitectura propia de MASMOVIL, esto conlleva el análisis de la arquitectura actual, así como el análisis y diseño de la solución a los requerimientos de negocio dentro del grupo, también se contempla como parte del proyecto la detección de oportunidades de mejora en dicha arquitectura. Las soluciones técnicas deben estar alineadas con el enfoque técnico de MASMOVIL que en líneas generales es el siguiente:

- Arquitectura dirigida por eventos. Mediante apache Kafka con mensajería en formato Apache Avro https://avro.apache.org/ .
- Desarrollo de microservicios y APIs en RxJava con Eclipse Vert.x https://vertx.io/ .
- Implementación de patrón SAGA de orquestación de microservicios mediante Uber Cadence https://cadenceworkflow.io/
- Instalación de componentes necesarios en cluster GCP (Google Cloud Platform) de MASMOVIL
- Construcción de aplicaciones usando Bazel https://bazel.build/ , herramienta de construcción de Google con las que construye, GMail, Google Search, Google Docs.

MASM-CTDR-2: Chaos Engineering para Testing en infraestructura cloud. El objetivo es mejorar la confianza en la resilencia de los sistemas distribuidos en infraestructura cloud gracias al uso del testing para predecir errores. Buscamos asegurar metodologicamente y a nivel de desarrollo todo el ciclo de vida alrededor del uso de infraestructuras en la nube basandose en forzar, precisamente, esos errores y esperar la recuperación sin intervención de los sistemas o incluso con procedimientos ya definidos y recreados en la validación. El proyecto profundizará en la disciplina del Chaos Engineering diseñando estrategias que consoliden la practica de definición, implementación, ejecución y validación de experimentos iterativos en equipos de desarrollo con cultura Devops. Partiremos de un escenario con sistemas distribuidos dentro de un cluster en un proveedor de infraestructura cloud (Google Cloud Platform) y abordaremos aspectos como:

- Uso de productos enfocados en Chaos Engineering (Chaos Mesh, Chaos Monkey, Gremlin, Litmus, ChaosBlade…)
- Integración de testing de infraestructura en ciclo de vida de desarrollo (Jenkins, ArgoCD
- Estudio de criticidad de escenarios a validar y diseño de plan de experiimentos/pruebas
- Criticidad y uso de herramientas de monitorización y desempeño en la estrategia
- Operación como código e integración en sistemas de alarmado
- Estrategias de mantenimiento de servicios de prediccion de errores
- Reporting de errores encontrados y trazabilidad en el ciclo de vida

MASM-CTDR-2: Creación de modelo de estimación del churn utilizando Vertex AI. El objetivo del aula consistiría en desarrollar y entrenar modelos de machine learning utilizando el framework de google cloud Vertex AI. Como datos de entrada pueden utilizarse todos los disponibles en el wharehouse de bigquery, previo paso a un proyecto nuevo y ocultación de información sensible. Especialmente se consideran interesantes los datos de las siguientes verticales: Ticketing, ATC, Servicios, Redes, Etc.

- La salida del modelo debe ser una estimación de la probabilidad de baja del cliente en un determinado plazo de tiempo. Deberá estudiarse cómo tratar las portabilidades internas (entre marcas del grupo).
- En cuanto a tecnologías/frameworks, para la realización de los modelos se utilizará Vertex AI de google cloud. Haciendo especial hincapié en la “industrialización” del proceso de creación y entrenamiento de los modelos.

MASM-CTDR-3: Estado del arte del ciclo de desarrollo y desempeño en las arquitecturas serverless. El objetivo es la construcción de una aplicación web sencilla con área pública y privada que nos permita:

- Analizar las funcionalidades disponibles y describir qué nos ofrece la arquitectura serverless bajo análisis para utilizar aquellas que sean las más adecuadas para la solución a implementar.
- Proponer diferentes patrones de arquitectura de renderizado web y performance de la aplicación en base a la arquitectura serverless que se adapte a la solución a implementar

- Evaluar la experiencia del desarrollador sobre dicha arquitectura, describiendo el ciclo de desarrollo, ventajas e inconvenientes que introduce
- Evaluación del desempeño de dicha web con la medición de web vitals que nos permita obtener una visión real de la experiencia de usuari
- Para el desarrollo de la aplicación se recomienda hacer uso de remix o nextjs. Las arquitecturas serverless que se evaluar

MASM-CTDR-3: Mapa interactivo con AR. El objetivo es la construcción de un ecosistema de aplicaciones apoyándose en la tecnología de Google para apps: Firebase. Se debe realizar una app que implemente la tecnología de backend de Google para controlar el login y el almacenamiento de los modelos (Firebase/Firestore). Una vez hecho login con Google, deberíamos poder visualizar en un mapa puntos de interés colocados por uno mismo u otros usuarios, dichos puntos se deberán poder visualizar (si son cercanos) utilizando el modo realidad aumentada, situando el modelo en el plano visual y marcando la distancia hacía la marca Cada punto tendrá una imagen y una descripción asociada a sus metadatos (usuario, coordenadas, fecha, etc). Se puede realizar tanto en iOS como en Android. Lo ideal sería que se desarrolle en conjunto con varias personas y que haya diversidad de plataformas si los recursos lo permiten.

- Tecnologías/frameworks Android: ARCore para Android, Kotlin (en lugar de java), Compose para la UI, Corrutinas
- Tecnologías/frameworks iOS: ARKit/RealityKit para iOS, Swift (en lugar de obj-c), SwiftUI para la UI, Combine