Entrevista En Diferido 22: César Izquierdo

12 de Diciembre de 2019 a las 00:53

Comenzamos una nueva entrevista con un invitado con perfil de desarrollador, su nombre es **César Izquierdo.**

Antes de todo, gracias por aceptar la invitación y apoyar a este proyecto. La primera pregunta es una presentación por parte del entrevistado, para que los lectores te puedan conocer un poco.

Entrevista En Diferido: ¿Te puedes presentar en unas líneas?

César Izquierdo: Muchas gracias a ti por invitarme. Me llamo César Izquierdo Tello. Soy de Valencia. Estudié Ingeniería Informática en la Universidad Politécnica de Valencia y desde casi 8 años estoy viviendo y trabajando en Holanda. Llevo en la industria del desarrollo desde el año 2006. Hasta el Octubre pasado estaba trabajando como desarrollador freelance, pero acepté una oferta de uno de mis clientes (ParkNow Group) para trabajar en plantilla como Senior Software Engineer.

EED: Para empezar , unas cuantas preguntas cortas para conocerte mejor y conocer tu entorno tecnológico.

¿Qué ordenador utilizas habitualmente?
¿Sistema operativo?
¿IDE o Editor con el que trabajas?
¿Lenguaje de programación?
¿Frontend o Backend?
¿Un paradigma de programación?
¿Framework de JavaScript?
¿JavaScript Vanilla o framework de JavaScript?

CI: ¿Qué ordenador utilizas habitualmente?
Cuando me hice freelance me compre un MacBook Pro 13 pulgadas de 2018 (no entro en las especificaciones ya que todo el mundo las sabe), que era el ordenador que llevaba utilizando como empleado en otra empresa en los últimos 4 años. Ahora en ParkNow tengo un MacBook Pro 15 pulgadas de 2018.

¿Sistema operativo?
Ninguna sorpresa ya que he dicho MacBook Pro.

¿IDE o Editor con el que trabajas?
Utilizo varios, depende en lo que este trabajando. Como uno de mis tareas de desarrollo actual es desarrollo de aplicaciones Android, utilizo Android Studio. Cuando tengo que desarrollar para frontend web o para backend utilizo Visual Studio Code.

¿Lenguaje de programación?
Ahora mismo: Kotlin para desarrollo de Android para el 100% de nuevas funcionalidades y refactorizaciones. Java por el mantenimiento de todo el código existente en las aplicaciones Android que desarrollo. Cuando hago desarrollo backend en ParkNow estoy haciéndolo en .Net, ya que es el lenguaje de programación predominante en esa empresa para backend.
Lenguajes en los que he desarrollado aplicaciones para empresas y algunos que conozco: Javascript, PHP, algo de Swift y algo de Python.

¿Frontend o Backend?
Ambos. Depende de la tarea y el momento. Durante mi carrera he desarrollado/mantenido muchos backend´s, API´s y me encanta. Pero también desarrollado algún frontend principalmente en React y me especializado bastante en desarrollo de aplicaciones móviles. Empecé a desarrollar aplicaciones móbiles cuando surgieron los frameworks para desarrollar aplicaciones híbridas, que estaban basados en Javascript (como Cordova, ionic) y acabe desarrollando aplicaciones nativas para Android. Como he desarrollado mucho frontend móvil me he obsesionado bastante con tener una API bien diseñada y que haga el desarrollo front mucho más ágil y eficiente. Me siento muy cómodo desarrollando en backend y es lo que ahora mismo mayoritariamente centro mi carga de trabajo y aprendizaje. Ahora mismo estoy muy centrado en desarrollar aplicaciones serverless en AWS.

¿Un paradigma de programación?
Programación orientada a objetos y programación funcional. Entiendo algunas de las problemáticas que tiene la OOP, entiendo que hayan opiniones en contra, pero a pesar de que haya voces que digan que OOP es un negocio que se nos ha vendido a los programadores, pienso que el OOP era el paso necesario en el avance de las técnicas de desarrollo de software y hacerlo accessible para todo el mundo. No utilizo un lenguaje de programación complemente funcional, pero lenguajes que estoy utilizando actualmente tienen varios grados de características de programación funcional (Kotlin, Javascript)

¿Framework de JavaScript?
Actualmente no. (React es una librería, por lo que no lo metería en esta categoría). He utilizado Vuejs para proyectos personales y me parece lo mejor.

¿JavaScript Vanilla o framework de JavaScript?
Depende. Muchas veces los proyectos vienen con restricciones de tiempo, dinero, por lo que un framework (si y solo si lo conoces) te puede aliviar de las tareas que normalmente se repiten una y otra vez. Además te pueden dar unas guías de como estructurar tus proyectos, modularizarlos para que su manteniendo sea más sencillo. Pero todo eso se
puede hacer sin framework. Además nadie dice que haya parte de tu código que este bajo
los estandares de tu framework y el resto en Vanilla (por ejemplo, la lógica de negocio). Al final detrás de ese framework sólo hay Javascript.

EED: Me imagino que mas de una vez , te han preguntado por algún recurso para aprender un lenguaje, framework o librería. Voy a poner

una pequeña lista con una serie perfiles que quieren aprender algo nuevo, me gustaría que me dijeras un par de recursos para cada uno de ellos.

- **Una persona, con cero conocimiento, que quiere aprender a programar.**
- **Desarrollador de C/C++ que quiere aprender desarrollo web.**
- **Desarrollador de frontend que quiere aprender desarrollar en Android.**
- **Desarrollador de Backend que quiere aprender Frontend.**
- **Programación Orientada a Objetos a Programación Funcional.**
- **Desarrollador de PHP que quiere aprender JavaScripts**

CI: Antes de contestar me gustaría comentar que ningún de los perfiles que mencionas tienen "realmente" cero conocimiento. Da igual en que desarrolles, hay áreas de lo que has aprendido que se pueden transferir a otras areas. Es algo que los desarrolladores mas experimentados dominan muy bien y que el resto de mortales tenemos que aprender a hacer. Programación es programación, por lo que hay cierta carga de conocimientos que te pueden ayudar a aprender otras técnicas como se dice en inglés ... "connecting the dots". Darte cuenta de esto, hace las transiciones de aprender nuevas cosas menos dolorosas :)
También me gustaría decir que los recursos que voy a mencionar son muchos que en cierto grado he probado y por lo tanto me voy a dejar un montón muy buenos. Siempre es bueno investigar cuales recursos nos pueden ir bien.

Desarrollador de C/C++ que quiere aprender desarrollo web.
Depende de lo que consideremos que es desarrollo web. Si con desarrollo web vemos el front y el back, puedo recomendar lo siguiente:
Generalidades:

- Los MOOC que hice en su dia han cambiado un poco, pero alguno que ofrezca una vision global como este por ejemplo: https://www.coursera.org/learn/html-css-javascript-for-web-developers puede estar muy bien
- Este para mi es uno de los mejores recursos para iniciarse en esas tecnologías web: https://www.freecodecamp.org/
- Lo anterior es para hacerse como una idea general de las tecnologías que puedes ir utilizando en web, pero desde mi punto de vista es fundamental que se entienda el ciclo de vida de una “request”, entender que pasa cuando accedemos atraves del navegador a un sitio web. Una introducción puede ser un video como este: https://www.youtube.com/watch?v=eesqK59rhGA
- Esta es una documentación que no pararía de recomendar: https://developer.mozilla.org/es/ Llena de recursos para aprender tecnologías web y ademas las APIs de referencia para Javascript. Alguna vez he contribuido como traductor de la documentacion técnica de Javascipt y la version en español esta muy bien.
- Finalmente, si lo que buscas es un roadmap con todos los pasos recomendados, para mi este es el mejor: https://github.com/kamranahmedse/developer-roadmap Creo que el roadmap es bastante acertado en las cosas que hay que aprender en la parte mas practica.

Desarrollador de frontend que quiere aprender desarrollar en Android.

- Aprender Kotlin es fundamental. Lo ideal seria que aprendiera primero Java, pero si quiere ahorrarse ese paso, puede ir directamente a Kotlin.
- Desde propia experiencia yo haría lo siguiente:
  - Este libro para aprender Kotlin suelto: https://www.amazon.com/-/es/Josh-Skeen/dp/0135161630
  - Este libro para aprender a desarrollar aplicaciones Android, donde se explican de una forma extraordinaria los conceptos básicos. Android Programming: The Big Nerd Ranch Guide. Si os logeis la ultima version los desarrollos los hacen directamente en Kotlin https://www.amazon.com/-/es/Bill-Phillips/dp/0135245125/ref=sr_1_1?__mk_es_US=%C3%85M%C3%85%C5%BD%C3%95%C3%91&keywords=Android+The+big+ne1_
  - Luego me haría los CodeLabs que Google ofrece para desarrollo de Android: https://codelabs.developers.google.com/
  - Luego me estudiaría la guia de Android oficial, que es super detallada. Ahi podras aprender muchas cosas relacionadas con el desarrollo Android y todos como la aplicación interactuar con todo el hardware del movil.
  - Recientemente me he leído este libro de Antonio Leiva que es una pasada: https://antonioleiva.com/kotlin-android-developers-book/ Kotlin for Android Developers.
  - Como recurso para hacerte una subscripción e ir haciendo pequeños cursos, este es genial: https://www.raywenderlich.com/

Desarrollador de Backend que quiere aprender Frontend.
Los mismos recursos que he mencionado anteriormente.

- Los recursos que he puesto en el punto anterior valdrian aquí.
- Me centraria en FreeCodeCamp con algún MooC sobre HTML5. CSS y Javascript.
- Si quieres tambien tiraria de libros sobre todo por la parte Javascript. -
  - Este libro que además es online y gratis me encanta https://eloquentjavascript.net/
  - Un poco mas avanzado: Javascript: The Good Parts https://www.amazon.com/-/es/Douglas-Crockford-ebook/dp/B0026OR2ZY
  - Para Javascript este es un desarrollador y creador de contenidos que me encanta: https://ericelliottjs.com/ Ademas de tener cursos online

Programación Orientada a Objetos a Programación Funcional.

- Sin duda este repositorio. https://github.com/leandrotk/functional-programming-learning-path Un "learning path” ya creado por otro desarrollador. Desde ahi podemos hacer nuestro propio learning path.

Desarrollador de PHP que quiere aprender JavaScript

- Ummm.... Creo que he podeis encontrar la mayoría que recomendaría en los apartados anteriores.

EED: Veo que conoces varios lenguajes y framework de programación , desde tu conocimiento y experiencia .

¿Cuál crees que es el mejor lenguaje para aprender a programar?
¿Qué lenguaje no recomienda para aprender a programar?

CI: Es una gran pregunta, por que para mí no existe el concepto de mejor lenguaje para aprender a programar.
Aprender a programar, desde mi opinión, es aprender a resolver problemas con código, pero no necesariamente con un lenguaje específico. Se puede aprender a programar con cualquier lenguaje de programación, es mas, es interesante resolver los mismos problemas en diferentes lenguajes.
En mi opinión es mejor centrarse en aprender fundamentos como algoritmos, estructuras de datos, técnicas de diseño de software y para la parte práctica, resolver katas con problemas con algoritmos, programas inventados, proyectos personales, estructuras de datos con el lenguaje que se quiere. Una técnica también interesante es coger un lenguaje y explorar su API escribiendo tests.
Si quisieras optimizar algo de tiempo, si sabes una dirección que te apetece probar en práctica real para el trabajo o futuro trabajo entonces te puedes poner como lenguaje de práctica, tu lenguaje target. Por ejemplo, si en tu empresa utilizan Python en backend y quieres convertirte en desarrollador de ese backend, lo suyo, es que las prácticas las hagas en Python y luego dediques otro tramo de tu tiempo a planificarte como aprender todo lo que esta relacionado con el área de backend. Pero insisto que es interesante resolver los mismos problemas en otros lenguajes.
El motivo es que cuando coges un poco de práctica, empiezas a ver muchos patrones que se repiten entre lenguajes, muchas veces las APIs de estos lenguajes son similares y puedes adaptarte mejor a posibles cambios.
Digamos que te conviertes en un desarrollador mas completo, o es lo opino.

EED: Pregunta fija en todas las entrevista, donde solo cambia el contexto. En tu caso, como eres desarrollador de Kotlin...

¿Qué añadirías a Kotlin?
¿Qué modificarías de Kotlin?
¿Qué eliminarías de Kotlin?

CI: Hay muchos esfuerzos para hacer Kotlin multiplatforma (https://kotlinlang.org/docs/reference/multiplatform.html). He hecho algunas pruebas y es prometedor. Usar kotlin para crear tu web api, o tu backend, tus aplicaciones mobiles o tu frontend. Además se pueden escribir aplicaciones serverless en Kotlin https://www.raywenderlich.com/5777183-write-an-aws-lambda-function-with-kotlin-and-micronaut
Para mí Kotlin tiene todo lo que un lenguaje moderno debería tener.

EED: Esta pregunta es de una persona que conoces , Daniel Primo. Me comento que estuvistes en un proyecto muy interesante, un concurso de startup o algo similar, relacionado con la ESA. Podrías comentar algo sobre ese proyecto.

¿De qué trataba ese proyecto?

CI:

Para los que no esten familiarizados, la ESA es la Agencia Espacial Europea. Cada año, empresas privadas que tienen inversiones en la agencia espacial y la propia ESA organizan un concurso en el que el objetivo es crear una startup en una semana. Esta startup tiene que solucionar un problema usando una aplicación móvil. Además, para resolver ese problema en la que se basa el negocio de la startup tiene que usar como fuente principal de datos, los datos de los satélites de la agencia europea. En concreto los Sentinel.

Nos eligieron a 24 personas, algunos hicieron la inscripcion directamente en equipo y otros como yo, individualmente, y nos llevaron una semana a Frascati (Italia), donde se encuentra uno de los observatorios de la ESA en Europa.

Desde la ESA lo que quieren conseguir con este tipo de eventos es acerca los datos que los satélites capturan para que puedan ser usados en aplicaciones reales y solucionar problemas, por lo que están haciendo una labor de acercamiento con la comunidad de desarrolladores. Digamos que es un hackathon, pero extendido a una semana y además de tener un prototipo de la parte tecnológica de la idea, tambien teníamos que tener la parte de negocio y explotación de la idea lista. Si ganábamos el concurso, tenias acceso directo a un programa de aceleración de startups patrocinado por la ESA.

Forme equipo con un chico polaco y dos alemanes. Fue una semana super intensa. Por un lado, intentar aprender conceptos relacionados con satélites espaciales, que al menos yo no tenia ni puñetera idea. Si queréis saber mas sobre las misiones Sentinel, podéis bajaros la app para Android, donde podéis ver los satélites a tiempo real https://play.google.com/store/apps/details?id=esa.sentinel&gl=NL
Cada satélite tiene una instrumentación diferente que captura imágenes terrestres que ofrece diferente información. Esta captura no la puedes tener a tiempo real por que hay una latencia desde que el satélite captura la imagen y la ESA la tiene disponible a través de su plataforma. Ademas no es a tiempo real, por que el satélite esta orbitando por lo que no pasa un punto otra vez, algunos en horas, otras hasta días o semanas. Las imágenes capturan una area determinada, por lo que no es posible simplemente obtener por una localización determinada, si no que tienes que componer el area deseada dependiendo de las imágenes disponibles cercanas. Hay mucho trabajo de procesamiento de imágenes, por que ademas de la propia información de la imagen tienes que hacer correcciones atmosféricas por que entre otras cosas, hay algunas imágenes que no se ven correctamente por que en el momento de la captura habían nubes. En fin, a nivel técnico muchas cosas que aprender en poquísimo tiempo y sin tener nociones previas de ese area, fue bastante difícil, frustrante casi todo el tiempo. La parte del negocio fue mas fácil (al menos para mi), una vez teníamos la idea de que queríamos hacer. Aunque para montar un plan de negocio son muchas cosas que tienes que tener en cuenta, en muchos casos, los pasos para llegar a tener un plan de negocio razonable y que convenza esta mucho mas definido. La clave en este caso es la idea. Encontrar una idea de escala y sostenible es bastante complicado. Teníamos suficiente para crear un negocio sostenible para unas pocas personas, pero no para crear la siguiente Tesla. Tecnológicamente estoy bastante orgullos por que con el tiempo tan limitado que teníamos y los conocimientos limitados del dominio, conseguimos integrar datos con ayuda de librerias de terceros que por ejemplo podían ofrecer los cambios de profundidad del nivel de un rio en un periodo de tiempo determinado.

Nuestra idea consistia en una herramienta de soporte para los capitanes de barcos de mercancías en los ríos. El transporte de mercancías por rio en europa es extenso, poco explotado. Aunque los capitanes tienen muchas herramientas de navegación tienen pocas de soporte de decisiones. Por ejemplo, les ayudábamos a planificar mejor su ruta, paradas que tenían que hacer, condiciones atmosféricas, posible estado del agua. Descubr

imos que los capitanes de estas embarcaciones, se hipotecan para poder tener estos barcos y llevan consigo a su familia durante la mayor parte de sus trayectos. Su barco es su casa, por lo que tomar mejor sus decisiones podría aliviarles de mucho sufrimiento logístico y hacer

mejor su trabajo. Ademas, planeamos hacer acuerdos con puertos, ya que el trafico en los puertos es un problema en muchos puertos.

Uno de los puntos débiles de nuestra idea fue el mercado. Calculamos un mercado potencial de 60.000 clientes en europa, con una facturación esperada de aprox. 1 millon anual. Lo que era muy poco comparado con los ganadores del concurso que su mercado (la pesca en las costas finlandesas) mueve actualmente mas de 500 millones de euros anuales.

Si alguno esta interesado, podéis leer mas sobre el camp y como mandar la solicitud para entrar en el concurso del 2020 aqui: https://www.app-camp.eu/

EED: Otra pregunta fija que suelo realizar principalmente a desarrolladores.
A continuación pondré una lista de herramientas, tareas o conceptos que pueden ser utilizados en el desarrollo de una aplicación, me gustaría que respondieras ,de forma breve, con el grado de importancia
que consideras que debe conocer un desarrollador . Por ejemplo, si es muy importante conocerlas o es algo secundario.

Depurador
Git
Testing
Patrones de diseño
Documentación
Linting
Profiler
Comentarios(en el código)
Framework
Base de datos
Docker
Despliegue

CI: Depurador:

El depurador es una herramienta fundamental. Esencial para el día a día del desarrollador. Nos ayuda a además de conocer el comportamiento de nuestro programa en ejecución, nos ayuda a entenderlo el código mejor. Pero me gustaría añadir que creo que para ser mejor desarrollador hay que intentar averiguar que va a hacer nuestro programa antes de ejecutarlo. Como práctica, si por ejemplo puedo aislar trozos de programa que voy a depurar, primero intento releer el código he intentar ponerme en el lugar
del depurador e intentar repasar mentalmente, que contendrán nuestras variables, cómo se comportarán nuestros programas.

Git

Git es una de las primeras herramientas que aprendería. Tiene muchos beneficios aprender como primeras herramientas. Muchas empresas en las que vamos a trabajar (por supuesto no todas) utilizan Git. Cualquier código que desarrolles estará en un gestor de versiones como Git, por lo que aprenderlo, saber utilizarlo correctamente y saber lo que se hace es fundamental. Y evitar usar las herramientas GUI de Git para aprenderlo. Es mejor aprenderlo desde línea de comandos, aprendiendo los principios, aprendiendo que estas haciendo. Yo cometí ese error, y en un trabajo mandé a producción una rama que tenía más de un año de antiguedad. :(

Testing

Fundamental. Todas nuestras nuevas features o refactors tienen que tener tests. Eso nos asegura que nuestro código hace lo que debe hacer. No hay excusas válidas para no hacerlos. Siempre se puede buscar la manera y el tiempo de hacerlos. Como dice el libro de Clean Code, los tests son código y como tal, hay que mantenerlo y refactorizarlos. Hay que cuidarlos como cualquier otra parte de nuestro código.

Patrones de diseño

Importantísimo si queremos hacer soluciones que perduren estables y escalables en tiempo. Pero cuidado con los patrones de diseño. Hay muchos patrones y cada uno se puede o no utilizar en determinadas situaciones. Además, los patrones no tienen que ir antes de la resolución del problema. En el diseño de una solución, hay unos pasos por los que pasa nuestro código, desde un estado en lo que nos importa es simplemente que funcione, hasta el momento que aportamos una solución robusta, fácil de ampliar y de cambiar, que es donde pueden entran los patrones de diseño.

Documentación

Es muy importante aprender a documentar bien, no tanto la cantidad de documentación. Tenemos que saber quienes van a ser los lectores de la documentación y documentar lo que realmente sea útil para esas personas. Hay que aprender a redactar mejor

Linting

Importante conocerlos y usarlos. Cuanto antes lo adaptemos, más fácil será obtener sus resultados.

Profiler

Lo mismo que el Linting.

Comentarios(en el código)

Sólo los necesarios. Sólo comentar lo que nuestro código por si mismo no pueda explicar bien.

Framework

Es una gran artefacto de software, pero tenemos que aprender cómo funciona. Nunca depender de él y conocer como funciona, para

modificarlo si fuera necesario o arreglar los errores que puedan surgir.

Base de datos

Importantísimo. Tenemos que aprender a trabajar con bases de datos, relacionales o no.

Docker

Es interesante que aprendamos como usarlo y como funciona, pero no creo que sea esencial en general. Muchas empresas lo están utilizando. También para proyectos personales. Solo lo priotarizaria en el caso que para mi trabajo me hiciera inmediata falta. Pero no dejaria que se quedara en el olvido he intentaria aprender los basicos cuanto antes. Muchas de las infraestructuras en el cloud actuales, están usando Docker.

Despliegue

Cada vez más la línea entre la persona que desarrolla y el que despliega es cada vez mas fina. Creo que hay aprender CI/CD en general. Pero yo empezaría a conocer sus principios y prácticas. Después de eso pasaría a herramientas específicas dependiendo del contexto. Al final muchas de ellas tienen las mismas funcionalidades por que se han creado bajo los mismos principios. Pero creo que no es super necesario en una primera etapa de desarrollador.

EED: En varias respuesta aparece la programación funcional, que parece que esta en auge, conozco un para desarrolladores que estan empezando con la programación funcional y en mi canal de Python he puesta varios recursos sobre programación funcional porque me estoy encontrando bastante recursos. Pero lo primero sería conocerla un poco, porque es algo diferente a lo que se encuentra un desarrollador habitualmente.

¿Qué es la programación funcional?
¿Qué ventajas y desventajas te has encontrado?

Muchas veces cuando aprendres un concepto nuevo de programación, mas que dificultad es si has alcanzado el nivel suficiente de programación para entender lo correctamente. Hablando de la programación funcional.

¿Cuándo se debe aprender programación funcional?

CI: No tengo una gran experiencia en programación funcional, pero voy a intentar explicar mis experiencias. Lo primero que me gustaría comentar es que, quien te diga que tienes que olvidarte de todo lo que has aprendido hasta ahora de programación para aprender programación funcional te esta engañando. A mí me gusta verlo como lo escuché en una charla donde el ponente explicaba que la programación orientada a objetos se había vuelto extremandamente complicada, con demasiados efectos secundarios, con demasiadas condiciones de funcionamiento.

Con la programación funcional no intentas tirar a la basura todos los conceptos de programación orientada a objetos pero lo que haces es despedazar la programación orientada a objetos y quedarte con cosas que funcionan añadiendo un nuevo paradigma para escribir código que sea más claro, más limpio, que puedas explicar con más facilidad, que no tenga a priori efectos secundarios.

El ponente de la charla comentaba que la programación funcional coge robado de las matemáticas el concepto de las funciones. Las funciones no como los procedimientos o métodos que conocemos del POO. Así, una función en matemáticas es una abstracción que relaciona elementos de dos conjuntos, en el que un elemento de un grupo le corresponde un elemento de otro grupo. Es decir, que para un input tenemos siempre un output. Así la programación funcional hace énfasis en las funciones, las cuales se basan en la regla que estas funcionan tienen en cuenta su input para producir un output. Pero para crear estas funciones, todavía estamos usando muchos de los elementos de la POO.

Cuando aprendí a programar un profesor mío para explicar que era un programa, hizo el símil de un programa como una receta de cocina. Si quieres hacer una tortilla, tienes que seguir unos pasos. ¿Realmente necesitamos modelar la realidad de una receta con objetos Receta, Paso, Ingrediente, etc? Lo que realmente estamos haciendo es ejecutar un paso, por ejemplo batir un huevo, que como input puede tener dos huevos y cómo output es una copia de los huevos ya batidos. Y así hasta acabar la receta. Podremos reemplazar las funciones de nuestra receta, componerlas de otra forma, con otros inputs, para hacer otras recetas y no hemos necesitado en ningún momento el concepto de objeto, solo datos y funciones.

La PF se basa también en la inmutabilidad, evitando efectos colaterales. Conseguimos organizar la realidad del problema, en pasos claros, predecibles. Obviamente, los programas reales, tienen accesso a API´s externas, a bases de datos y a ficheros. Y también en algunos casos, necesitamos algún sistema que nos controle el estado de ciertos datos de nuestro sistema. Lo que en POO llamaríamos lógica de negocio, en PF, algunos lo llaman efectos colaterales. Cada lenguaje funcional lo maneja de forma diferente, pero la idea es que tienen abstracciones concretas que establecen puentes entre las funciones y los efectos colaterales. Algo que en POO, cuesta bastante y aunque lo intentemos encerrando el acceso a estos efectos mediante capas, en muchas ocasiones, nuestros objetos se llenan de código que entremezcla los efectos secundatios con el resultado de nuestros métodos. En mi opinión, en la PF esto se vuelve más natural, con menos artefactos.

No creo que haya un momento para aprender programación funcional. Exponerse a nuevos conceptos sólo nos hace mejores, nos hace tener la mente más abierta. También nos ayuda a trasladar conceptos de unos sitios a otros. Nadie dice que te hagas un experto y te olvides del resto de cosas. Exponte cuanto antes, valóralo por ti mismo, intenta escribir el mismo código de algún programa que tengas en PF, compara y analiza.

EED: Cuando una empresa quiere contratar un desarrollador, necesita mirar muchos aspectos de los candidatos; titulación, experiencia, actitudes... Es un proceso complejo porque quieren acertar en la elección del candiatos, pero en la actualidad existen herramientas como GitLab o GitHub donde se puede ver el código y su calidad de un desarrollador. De esta manera, conocer muchas habilidades que en un proceso de selección es mas complicado detectar. Desde tu experiencia y conocimiento.

¿Cómo valoran la titulación en informática y tus repositorios en GitHub o similar, en un proceso de selección?

CI:

Titulación académica.

Creo que se valora, te puede llegar a destacar de otros candidatos, ya que se entiende que al menos se te ha dado una formación en informática básica (otra cosa muy distinta es que realmente sepas). Pero no es crítico. Muchas empresas están contratando programadores que aprendieron a programar por su cuenta o por bootcamps. En mi primer trabajo de programador aún no tenía la titulación. Las cosas están cambiando y al menos ni para la negociación de un salario o para que te contraten, la titulación no es tan relevante (no en todas las empresas). De todas formas, la formación académica de cualquier tipo la recomiendo.

Experiencia.

Esta parte es una de las más importante para tener al menos la primera entrevista. Cuando hablo de experiencia, es del tipo que es relevante para el puesto. Si quieres acceder a un puesto, digamos, de Backend developer (que no sea Junior o Starter) tienes que tener experiencia relevante en Backend development. Esta experiencia a mí al menos me la han valorado de dos formas: todos los proyectos en los que he participado desarrollando a lo que aspiro y por otro lado proyectos personales, repos, aportaciones a open source que tengan que ver también con el puesto que aspiro.

Para mí el CV con la lista de empresas que has trabajado, ¨esta muerto¨ (en nuestro sector). Lo importante son los proyectos que has participado. En una misma empresa puedes haber trabajado en muchos proyectos backend y explicar que tipo de proyectos eran, tu rol en esos proyectos, el stack que usabas, es lo que realmente desde mi opinión se valora más. Yo cambiaria algunos CV´s de orientación a empresas a orientación a proyectos.

Estos dos aspectos anteriores son los que realmente te pueden hacer entrar en el proceso de selección real. Si te llaman, lo más probable es que haya un proceso de entrevista técnico. En todas partes no hay pruebas técnicas, pero la empresa que tiene un equipo técnico fuerte, generalmente hace pruebas técnicas, por que se quiere asegurar que la persona a que contrata les demuestra
lo que sabe. Muchos dicen que las pruebas técnicas están mal planteadas y en algunas ocasiones no reflejan todo lo que sabes (y en cierta medida estoy de acuerdo). Pero al final, esa prueba te da la entrada para que en otras entrevistas digas o demuestres que sabes más, asi que hasta que cambie la cosa, hay que preparárselas bien. Las pruebas técnicas las uso para conocerme mejor y para saber puntos de mejora.

En mi carrera las pruebas técnicas han tenido diferentes estilos: te mando un ejercicio en el que tenía que hacer un programa y me lo devuelves hecho en 24 horas, una prueba online, generalmente relacionado con una prueba de hacer un programa que tenía un gran contenido de algoritmos, recursividad, etc. o directamente una entrevista técnica hablada con un líder técnico de la empresa. En mi carrera he fallado tremendamente en muchas de esas pruebas, y me he dado cuenta que no le he prestado tiempo a prepararlas. Existen multitud de recursos y libros, que te ayudan a prepararlas. Si te mandan un ejercicio para que hagas en un día, lo importante es ceñirte a lo que te dicen. Hacer lo que te dicen con lo mayor calidad posible y si quieres meterle más azúcar, hacerlo solo si tienes tiempo. Por ejemplo, en una prueba para un trabajo de Senior Android Engineer, me mandaron para hacer una app pequeña en 2 días. Como no me daban UI, me centré demasiado en intentar hacer una UI de lujo, para que cuando usaran la app vieran lo bien que se veía y fallé en intentar demostrar lo que sabía de arquitectura, gestión de errores, organización por capas, diseño. Eso me hizo darme de cuenta que había aspectos técnicos que tenía que mejorar. □□

EED: Hablando de habilidades de un desarrollador.

¿Cuáles crees que deben ser las habilidades de un buen desarrollador?

CI: Ummm... pues para mí la lista sería en lineas generales, así:

- Responsable. En el sentido que en la medida de lo posible está haciendo lo que tiene que hacer, sabe por que esta trabajando y es responsable de sus tareas y de las cosas que tiene que hacer.
- Honesto. Aunque el ego es a veces fuerte, sabe decir abiertamente cuando sabe algo y cuando no, por que la fama o el reconocimiento no le importa, lo que le importa es aportar y aprender. Si no puede aportar algo, lo dice y así aprende de otros.
- Resolutivo. Los buenos programadores/profesionales son resolutivos. Tienen un problema entre manos y aunque en un primer momento no saben resolverlo, buscan respuestas. Saben buscar respuestas y entregan.Los buenos programadores entregan código constantemente.
- Estudian fundamentos. Los buenos programadores se preocupan en profundizar no sólo en la practica de programación si no en los principios que hay detrás. Saben que tener unos fundamentos sólidos los hacen mejores. No se obsesiosan con el acto de programar, se obsesisan con indagar el por que de las cosas y como hacer esos programas un poquito mejor. Saben que para eso no hay mas remedio que estudiar lo que se pueda.
- Estudian el trabajo de otros. Creo que los buenos programadores saben que por sus propios sesgos, necesitan nutrirse de ideas de otras personas. Por lo que escuchan, leen y estudian el trabajo de otras personas e incorporan cosas de esas personas a sus propias técnicas.
- Independientemente de como se sientan, tienen un cierto grado alto de determinación. Quieren ser buenos profesionales y se mueven hacia esa dirección. Intentando evitar distracciones que les desvíe de ese objetivo.

Esta cualidad claramente se puede volver negativa, pero pienso que en cierto grado es necesaria,

por que la programación o el mundo profesional en general, sufre un montón de altibajos. Todos los días no estamos al máximo rendimiento, hay semanas que no has aprendido nada. Las cosas a veces no salen como quieres, hay muchas otras obligaciones que atender. Pero incluso cuando todo falla la determinacion aflora para darles un empujón de motivación para continuar. 😊 □□

EED: Como última pregunta de la entrevista , una pregunta un poco diferente.

¿Qué te hubiera gustado que te preguntase? Evidentemente la respuesta será tuya.

CI: La verdad es que no se me ocurre muchas otras preguntas. Hemos tocado muchos frentes y eso siempre mola. A lo mejor me hubiera gustado que me preguntaras sobre mis proyectos actuales o sobre las cosas que tengo en mente aprender en un futuro. A la primera pregunta ahora mismo estoy involucrado en mi empresa en un proyecto en el que estamos evaluando la posibilidad de usar GraphQL como APIs para las aplicaciones móviles, web con los microservicios. A la segunda pregunta, estoy estudiando bastante sobre serverless en AWS y en breve me quiero examinar de la primera certificación. Quiero abrir mi propio blog para escribir sobre ingeniería y programación. Cómo experimento he abierto un repositorio en GitHub que quiero que me sirva como un cajón donde meter todo lo que aprendo de programación y tecnología (los recursos que he usado, programas que he hecho, apuntes propios, etc).

EED: Hoy es el último día de la entrevista y es el momento de la despedida, pero antes me gustaría agradecerte tu participación, espero que haya sido una experiencia interesante y entretenida para tí.

Puedes indicar tus métodos de contacto y si tienes algún proyecto, web, podcast o evento que quieras promocionar tienes este espacio para hacerlo.

Por último, me gustaría que me propusieras a una persona que tú creas que estaría dispuesto a participar en una futura entrevista.

Ha sido un placer , hasta la próxima.

CI: Muchas gracias por haberme invitado. Ha sido una experiencia muy entretenida y divertida. Espero haber aportado algo de valor a tus lectures.

Estos son los métodos de contacto que más utilizo y es donde la gente me puede encontrar más fácilmente:

Email: cesiztel@gmail.com
Twitter: @cesiztel

Cualquier cosa que os pueda ayudar, no dudéis en contactar. 😀□□□

La persona que me gustaría proponer se llama Moisés. Es de mis mejores amigos y además es Software Architect en una conocida consultora española. Tiene un perfil de desarrollador y diseñador de software muy fuerte, y puede aportar cosas interesantes.

Un placer 😄

Doy por finalizada la entrevista, la siguiente será el 16 de Diciembre.

Disponible una cuenta de Twitter para estar informado de las entrevistas y entrevistados. Twitter @EDiferido