Entrevista En Diferido 27: Juan Luis Cano y Marta Gómez Macías

17 de Mayo de 2020 a las 02:29

Comenzamos una nueva entrevista y volvemos con el formato de entrevista doble, los dos entrevistado tienen un perfil de programadores en Python y muy activos en esa comunidad. Sus nombres son Juan Luis Cano y Marta Gómez Macías.

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

Entrevista En Diferido¿Os podéis presentar en unas líneas?

Juan Luis Cano: ¡Buenos días! □□

Me llamo Juan Luis Cano, soy ingeniero aeronáutico por la Politécnica de Madrid y actualmente trabajo como Mission Planning & Execution Engineer en Satellogic, una startup argentina que quiere remapear el mundo construyendo una constelación de satélites. Estamos un poco locos y lo hacemos todo en Python □□□

Soy miembro fundador de la asociación Python España, que he presidido hasta desde 2014 hasta 2020, y organizador de las primeras 7 ediciones de la PyConES.

Y aparte de esto soy profesor asociado en el Instituto Empresa, publico software de código abierto, hago proyectos freelance... Y en el tiempo que me sobra escucho muchísima música, monto en bici y viajo 😊

Marta Gómez Macías: ¡Buenos días!

Me llamo Marta Gómez, estudié el grado en ingeniería informática en la UGR y actualmente trabajo como ingeniera software en VirusTotal, dónde usamos Python para nuestros microservicios.

Soy organizadora en YesWeTech, una asociación que promueve la tecnología en mujeres y niñas y da visibilidad a las mujeres de la industria. Solemos hacer talleres, open spaces, charlas, etc en Málaga y Granada!

En mi tiempo libre me gusta hacer katas de Codewars, la fotografía, escuchar música y hace poco he empezado a patinar :).

EED: Para empezar , unas cuantas preguntas cortas para conoceros mejor a

vosotros y vuestros entornos tecnológico.

¿Qué ordenador utilizas habitualmente?
¿Sistema operativo?
¿Lenguajes de programación que utilizáis?
¿Entorno multimonitor?
¿Editor o IDE?
¿Trabajáis en remoto o en oficina?
¿Que herramienta es imprescindible en vuestros desarrollos?

JLC:

1. Mountain Nickel i5 de 2017 (+ Slimbook Excálibur i5 de 2017 como backup)
2. Linux Mint 19.3 en todas partes
3. Python
4. Las 14" de mi Mountain
5. PyCharm, vim sin personalizar, Jupyter notebook
6. En remoto
7. git, tmux, mypy, pytest, black, papel + lápiz!

MG:

1. MacBook Pro 13 en el trabajo y Dell XPS 13 en casa
2. Mac OS X en el trabajo, Windows + WSL en casa
3. Python y Go mostly
4. No
5. Editor, vscode
6. Oficina
7. Vscode, pytest, git, papel y lápiz :)

EED: La siguiente pregunta es fija en todas las entrevistas ,solo cambia el contexto en función del entrevistado. En vuestro caso, esta claro el contexto.

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

JLC:

1. Un método limpio y unificado para distribuir aplicaciones (tipo pyinstaller, pero más simple y robusto)
2. dataclass (ya está attrs!) y todas esas cosas de la stdlib que se usan muy poco y perfectamente se podrían instalar con pip
3. Mejor soporte para stubs (.pyi) de MyPy y así no tener que declarar los tipos en el

mismo código

MG:

1. Añadiría requests a la librería estándar
2. Los ordered dicts
3. Las clases, para que tuvieran parte privada y pública

EED: En este año se ha dejado de dar soporte para Python 2 , finalizando el proceso de migración a Python 3.

¿Qué opináis sobre como se ha desarrollado ese proceso?

¿Cómo os gustaría que fuera el proceso de migración de Python 3 a 4?

JLC:

1. Opino que se ha desarrollado mal. Por un lado, los core developers subestimaron el esfuerzo que llevaría a la gente migrar el código (Python estaba más implantado de lo que pensaban, me parece). Por otro, las versiones 3.0, 3.1 y 3.2 tenían bastantes fallos, lo cual provocó una falta de confianza y retrasó aún más la migración. La 3.3 ya era usable, pero fue a partir de la 3.5 cuando realmente empezó a merecer la pena y la gente se puso a migrar. Finalmente, la estrategia de usar 2to3 fue claramente errónea, y se tendría que haber dado una forma a la gente desde el principio para que el código corriese a la vez en 2 y 3, pero six y futurize se empezaron a popularizar tiempo después. Pero en fin, es 2020, ya estamos aquí, y solo puedo alegrarme de que por fin hayamos dejado Python 2 atrás Creo que dar soporte de Python 2 a empresas gigantes es un mercado de muchos millones...

2. De alguna forma el proceso ya ha empezado. Algunos PEP introdujeron nuevos imports __future__, como el PEP 563, y no usarlos empezará a producir warnings en 3.9. En 3.10 se eliminará asyncio.coroutine() (la forma antigua de declarar corrutinas). A la gente no se le va a olvidar fácilmente el trauma de Python 2, o sea que si se rompe código otra vez en 4.0 no va a sentar bien. Yo daría en 3.12 un flag --strict o --enable-4 o similar que permitiese a los devs ver si su código se romperá en el futuro, que en 3.13 no usar este flag tirase un warning bastante ruidoso, y que 4.0 se promocione como "non-production ready" hasta que se solucionen los problemas más graves en 4.1.

MG:

1. Por parte del lenguaje de programación creo que lo han hecho bien. La depreciación llevaba anunciada con tiempo suficiente como para planear una migración y además se

ha estado dando soporte durante todo este tiempo. Hay documentación y scripts automáticos para migrar. El problema ha sido por parte de las empresas/personas que han dejado sus migraciones para última hora. Otro gran problema han sido las dependencias, muchas de ellas no han migrado a Python 3 haciendo imposible migrar los proyectos que dependen de ellas.

2. Sinceramente no se me ocurre una forma mejor para hacerlo.

EED: Uno de los ámbitos donde se utiliza bastante Python es en el campo de la Inteligencia Artificial, donde encontramos diversas herramientas de Deep Learning y Machine Learning, que permiten desarrollar una IA más "inteligente' capaces de realizar tareas más complejas.

Los dos sois programadores y entendéis perfectamente el proceso de desarrollo de un programa informático, la programación es muy compleja,requiere tanto conocimiento como práctica, e incluso, en mi opinión, un componente "artístico".

¿Creéis ,o no, que existirá una IA que programe mejor que un humano?¿Y cuando?

MG: En este caso hay que diferenciar entre diseñar un algoritmo y programar. Yo creo que el proceso de diseñar un algoritmo es bastante complejo no solo de realizar sino de definir de una forma que sea procesable por una IA. En cambio, sí que veo factible el traducir un algoritmo a alto nivel a código Python. En el fondo seguiría programando el humano, pero a un nivel de abstracción mayor.

JLC: Estoy totalmente de acuerdo con Marta. Por otro lado, siempre suelo ser muy escéptico cuando veo noticias sobre avances en IA porque, aun siendo espectaculares, detecto que se tiende a exagerar. Dos ejemplos recientes: la IA que derrotó al campeón de Starcraft 2 jugaba a velocidades muy superiores a las de cualquier humano, con lo cual no tenía ningún mérito **https://blog.usejournal.com/an-analysis-on-how-deepminds-starcraft-2-ai-s-superhuman-speed-could-be-a-band-aid-fix-for-the-1702fb8344d6** Por otro lado, un reciente estudio ha puesto en duda la efectividad del deep learning para clasificar mamografías **https://www.jacr.org/article/S1546-1440(20)30028-4/fulltext** Estos problemas se enmarcan en uno más grande que es la crisis actual de reproducibilidad de la ciencia.

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, en vuestro caso relacionado con Python. Responder ,de forma breve, con el grado de importancia que consideraís que debe

conocer un programador . Por ejemplo, si es muy importante conocerlas, secundario o no es necesario .

Programación Orientada a Objetos.
Git y similares
Entornos virtuales de programación(virtualenv,conda,venv..)
Patrones de diseño
Programación asincróna
Jupyter
Depurador
Panda y similares
Testing (Unittestt,Pytest...)
Despliegue de aplicaciones
Metaprogramación

JLC:

Programación Orientada a Objetos: importante
Git y similares: esencial
Entornos virtuales de programación: esencial
Patrones de diseño: secundario
Programación asíncrona: innecesario para data science, importante para el resto
Jupyter: importante para data science, secundario para el resto
Depurador: importante
pandas y similares: esencial para data science, secundario para el resto
testing (pytest!): importante
despliegue de aplicaciones: importante
metaprogramación: innecesario (en mi opinión afea el código y encima no está soportado en mypyc ni en micropython)

MG:

Del 1 al 10:

Programación orientada a objetos: 9.5. Conceptos muy básicos que se usan en el día a día como programador.

Git y similares: 6. Solo imprescindible en entornos profesionales y trabajo en equipo.

Entornos virtuales de programación: 6. Solo imprescindible si trabajas en muchos proyectos a la vez/cambias de proyecto cada poco tiempo.

Patrones de diseño: 9. Muy importante, no solo aprenderlos sino acostumbrarse a usarlos.

Programación asíncrona: 7. Es un concepto avanzado pero muy usado a día de hoy.

Jupyter: 6. Imprescindible si eres estudiante o científico.

Depurador: 8. Ayuda a aprender, ahorra tiempo arreglando bugs. Es muy útil.

Panda y similares: 5. Sólo si trabajas en data science/engineering.

Testing: 7. Empieza a ser importante en proyectos grandes aunque es bueno acostumbrarse a testear siempre.

Despliegue: 5. Sólo en entornos profesionales o si estás aprendiendo cloud.

Metaprogramación: 5. Concepto muy avanzado pero muy útil para algunas situaciones.

EED: Esta pregunta es de **José Carlos García**.

Como programadores de Python.

¿Qué es para ti lo mejor y lo peor de Python?

MG:

Lo mejor: la rapidez de desarrollo: en poco tiempo puedes hacer un prototipo para un proyecto. Es un lenguaje muy ágil y muy fácil de aprender. También me encanta el soporte que tiene para hacer programación funcional.

Lo peor: trabajar con hebras. Sin duda.

JLC:

Lo mejor: Sirve para todo (y ese "todo" cada vez abarca más), la sintaxis básica se aprende fácil comparado con otros lenguajes, su comunidad es ❤

Lo peor: La instalación, tanto del lenguaje en sí como de los paquetes. pip y setuptools

EED: Los dos sois ingenieros y próximamente será el 8M, siempre me he hecho una pregunta y no se la respuesta.

¿Porqué hay tan pocas mujeres estudiando ingeniería?

MG: Es una pregunta muy complicada y que mucha gente lleva preguntándose mucho tiempo. Hay datos y estudios al respecto, no solo de universidades internacionales sino también de universidades españolas.

Por un lado, está el problema de no tener mujeres estudiando informática. En mi opinión, esto se da por una mezcla de muchos factores: la gran cantidad de estereotipos que hay al respecto, desinformación, falta de apoyo o desmoralización por parte de docentes y familiares (de verdad quieres estudiar eso? no te gusta más una carrera de chicas? medicina?), y un largo etcétera.

Por otro lado, está el problema de mujeres que dejan la industria. Cuántas mujeres con más de 10 años de experiencia conoces en la industria? En este caso, hay muchos factores pero el principal suele ser el ambiente hostil al que las mujeres se enfrentan por ser una minoría (comentarios fuera de lugar por parte de compañeros, sentir que no encajas con el resto, que esto no es para ti, etc etc).

JLC: Efectivamente es complicado dar una respuesta rápida, pero para mí claramente hay un problema cultural: a los chicos se nos "empuja" a hacer determinadas actividades y no otras, y así pasa con las chicas. Es algo que me irrita particularmente porque las discusiones al respecto están plagadas de lugares comunes.

Es un hecho empírico que a las mujeres en Occidente les interesa menos la informática, abundan los datos sobre ello. Lo que mucha gente se apresura a decir sin ninguna base es que son motivos biológicos o genéticos, cosa que se ha desmontado en multitud de ocasiones. Lo que hacían nuestros antepasados neandertales no influye en nada, y lo que sí influye son los comentarios, las situaciones incómodas, los estereotipos y la falta de igualdad de roles.

Desde Python España llevamos 5 años promocionando actividades específicamente para mujeres, como los talleres de Django Girls y los meetups de PyLadies, y se observa todas las veces que cuando se promociona un evento para mujeres, acuden _muchísimas_ mujeres. Así que están ahí, solo que los hombres lo hacemos muy mal para que se sientan bienvenidas en espacios mixtos.

Con esfuerzo, concienciación, discriminación positiva que corrija las desigualdades estructurales y mucha escucha y empatía, conseguiremos que el porcentaje de mujeres en ingeniería sea representativo de la sociedad en su conjunto (es decir, un 50 %) y de paso que los hombres no tengamos miedo a bailar, cantar, llorar, pintar, coser, o expresar nuestros sentimientos y debilidades de cualquier forma sin ser juzgados por ello.

EED: Los dos trabajáis en campos diferentes de desarrollo, Juan Luis desarrollas aplicaciones científicas , Marta desarrollas herramientas de seguridad. Si una persona os pide consejo porque le interesa vuestro campo y quiere empezar en ese ámbito de desarrollo.

¿Qué recursos de aprendizaje(cursos,libros,videos,master...etc), le recomendarais para empezar?
¿Qué herramientas le aconsejarías que aprendieran?

MG: En mi trabajo hace falta una skill principal que es el saber desarrollar software y una secundaria que es saber sobre seguridad, esta última es necesaria para no implementar a ciegas sino tener contexto de la importancia y utilidad de lo que se está implementando.

En mi caso, he aprendido sobre programación en la universidad y un poco por mi cuenta también (en conferencias, tutoriales, etc). Sobre seguridad he aprendido sobre todo en mi trabajo, y hace poco hice un curso experto en la UMA sobre análisis de malware. Así que realmente no puedo recomendar libros como tal. Sobre herramientas, recomendaría Python para programar y en seguridad recomiendo probar un poco de todo porque es un campo tan amplio que realmente no basta con una única herramienta. En mi anterior trabajo trabajamos con SIEM y en el actual, con malware. Son cosas distintas pero ambas son seguridad.

JLC: El desarrollo de software científico es muy interdisciplinar, lo cual en cristiano significa que los científicos no saben programar y que los programadores no entienden las ecuaciones 😄 Así que si uno es capaz de tender un puente entre estos dos mundos el éxito está asegurado. En mi caso soy aeronáutico y aprendí a programar por mi cuenta, más allá de una asignatura de FORTRAN 90 en la Universidad.

Cosas que importan mucho a la hora de entrar en este mundo son no asustarse al trabajar con software compilado (instalar software científico es un poco pesadilla), tener una buena base matemática (álgebra lineal, estadística y cálculo numérico sobre todo) y ser una persona metódica en el desarrollo de software (no es tan fácil definir qué es un "test unitario" en un código de simulación, pero igual hace falta algún tipo de validación). Lo mínimo mínimo, tenerle respeto a la aritmética de punto flotante 😄 http://puntoflotante.org/ (esta web la traduje yo en su día y contiene varios enlaces)

Siguiendo con el autobombo, tengo un curso en YouTube gratuito y sin anuncios de Python para científicos e ingenieros (primera parte https://www.youtube.com/watch?v=ox09Jko1ErM&list=PLGBbVX_WvN7bMwYe7wWV5TZt1a58jTggB y segunda parte https://www.youtube.com/watch?v=nYtNq2D-new&list=PLGBbVX_WvN7as_DnOGcpkSsUyXB1G_wqb). En cuanto a libros, yo estudié con los libros de Carlos Vázquez Espí y Juan Antonio Hernández Ramos, y "A Primer on Scientific Programming with Python" de Hans Petter Langtangen es altamente

recomendable.

Los másters de ingeniería matemática en general son los que profundizan en este tipo de cuestiones, pero no tengo recomendaciones específicas.

EED: Los dos sois ingenieros con experiencia trabajando como desarrolladores , me imagino que conocéis el mercado laboral y sus necesidades.

¿Qué grado de importancia ha tenido el conocimiento adquirido en la ingeniería que habéis cursado , en vuestro trabajo actual?

Teniendo en cuenta vuestra experiencia laboral.
¿Cambiaríais la forma en que se enseña una ingeniería en la universidad?

JLC: En mis primeras experiencias laborales estaba bastante frustrado porque me daba la sensación de que lo aprendido en la Universidad me estaba resultando inservible, pero lo cierto es que, con el peculiar estilo pedagógico que se estila en nuestro país, sí que desarrollé una habilidad para investigar, buscarme la vida y enfocar problemas de diferentes maneras. Actualmente por suerte en mi día a día aplico conocimientos de cálculo numérico, mecánica celeste, geodesia... Así que estoy moderadamente satisfecho ☺

Cambiaría muchas cosas, pero sobre todo en universidades masificadas como las madrileñas reduciría drásticamente el número de alumnos por clase, y cuidaría el uso excesivo de las diapositivas, que en muchos casos no son un buen sustituto a hacer un desarrollo en una pizarra. También incidiría más sobre las habilidades blandas: trabajar en grupo, comunicar, visualizar, empatizar, gestionar, tolerar la frustración.

Por lo demás, creo que adquirir conocimientos teóricos debe seguir siendo fundamental en la Universidad, porque nos hace más libres y nos ata menos a "lo que el mercado demanda", que hoy es una cosa y mañana es otra totalmente diferente.

MG: Estoy completamente de acuerdo con Juan Luis. En mi experiencia, los conocimientos y habilidades que he adquirido en la universidad han sido la base que me ha permitido seguir creciendo y aprendiendo yo sola. En esta profesión nunca se deja de aprender, y eso es algo en lo que la universidad te prepara bien.

Respecto a cosas que cambiar también coincido con Juan Luis. Añadiría, el reducir el número de trabajos prácticos pero haciendo que cada trabajo enseñe varios conceptos. Cuando estudiaba en la universidad estaba desbordada a trabajos, hubiera preferido hacer uno más grande que englobara más cosas. Además eso también viene bien para aprender cómo gestionar un proyecto más grande, dividir un proyecto en tareas, planificar, etc.

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

¿Qué te hubiera gustado que te preguntase? Evidentemente , debes responder a tu propia pregunta.

MG:

¿Por qué decidiste hacerte ingeniera software? ¿Qué camino te ha llevado hasta dónde estás hoy profesionalmente?

Desde pequeña me han llamado mucho la atención los ordenadores, mis padres siempre dicen que aprendí antes a usar un ordenador que a hablar :). Siempre he tenido muy claro que me quería dedicar a algo relacionado con los ordenadores así que estudié ingeniería informática en la universidad.

Una vez ya en la universidad pude ver cada una de las posibles salidas profesionales que había y me decidí por la seguridad informática, pues me parecía una de las que más impacto podía tener y de las más emocionantes. También me gusta mucho programar, así que finalmente decidí mezclar las dos cosas. Lo bueno de la ingeniería software es que cualquier otra rama necesita de un software para agilizar su trabajo así que en realidad puedes dedicarte a cualquier campo que te llame la atención: medicina, finanzas, seguridad...

En una feria de empleo de mi facultad vino una startup en la que desarrollaban un producto de seguridad open source llamado Wazuh, y lo vi claro! :) Me uní al equipo al terminar mis estudios y he estado con ellos hasta mayo del año pasado, que me cambié a VirusTotal.

Me he encontrado con inseguridades y con dudas a lo largo del camino, pero siempre he sido fiel a lo que creía mejor y a lo que me gustaba. No ha sido fácil pero estoy muy contenta de estar dónde estoy hoy!

JLC:

¿Por qué es importante para ti el código abierto?

Me encanta que me hagas esa pregunta 😝

Aprendí a programar en ActionScript 3 (Flash) gracias a la comunidad Cristalab (que ya no está activa) y en PHP gracias a unos tutoriales publicados en los foros de 110mb, el hosting que yo usaba en su día y que ya tampoco existe. Personas que no conocía de

nada criticaron mis diseños, resolvieron mis dudas, y en un momento dado dieron la bienvenida a mis contribuciones.

El código abierto en particular y la cultura libre en general son lo que más se parecen al mundo en el que me gustaría vivir. Wikipedia ya fue una revolución pero aún nos queda mucho trabajo por hacer, mucho conocimiento por compartir, mucho software por liberar, y muchos libros abiertos por escribir. Me gustaría que cualquier persona en cualquier parte del mundo con una conexión a Internet pudiese aprender de cualquier cosa, sin barreras, costes añadidos, o necesidad de software carísimo.

Solo en el ámbito espacial, proyectos que hace años parecían una locura, como los nanosatélites, los picosatélites o los femtosatélites, hoy son una realidad gracias a que hay estándares abiertos, diseños que se pueden imprimir en plástico, y una comunidad creciente de entusiastas compartiendo aciertos y errores.

Nunca sabes quién o en qué lugar va a usar el conocimiento que ponemos en Internet disponible abiertamente. Y eso es maravilloso 😊

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

Podeís indicar vuestro métodos de contacto y si tienes algún proyecto, web, podcast o evento que quieras promocionar teneís este espacio disponible.

Por último, me gustaría que me propusieraís ,cada uno, a una persona que creaís que estaría dispuesto a participar en una futura entrevista.

Ha sido un placer , hasta la próxima.

MG: Muchas gracias a ti por la entrevista! Ha sido una experiencia maravillosa.

Sobre métodos de contacto, me podéis encontrar en Twitter (@mrs_darkdonado) o LinkedIn (Marta Gómez Macías).

Y sobre proyectos interesantes, soy una de las organizadoras de yes we tech: una asociación de mujeres en tecnología que está en Málaga y Granada. Organizamos talleres, charlas, open spaces, etc para acercar la tecnología a las mujeres y también colaboramos en otros eventos. Si os interesa podéis entrar en yeswetech.org :).

Y respecto a alguien a quien nominar, nomino a @alberpilot

JLC: ¡Muchas gracias Jose por las preguntas, a José Carlos y Andros por

recomendarme y a las y los lectores!

Las únicas redes sociales en las que estoy son LinkedIn y Twitter , ¡estaré encantado de conectar por ahí! Mi email es hello@juanlu.space por si alguien quiere mandarme un correo 😊

Seguid de cerca el trabajo de la asociación Python España, mandad charla a la PyConES 2020 de Granada https://2020.es.pycon.org/ y si no hay un grupo local en vuestra ciudad, pedidles ayuda y consejo para montarlo □□

Nomino a @EllaQuimica, química, pythonista y viajera □