**نود و هفت چیزی که هر برنامه‌نویسی باید بداند: ویژگی‌های یک API با طراحی مناسب**

[**https://programming.tosinso.com/fa/articles/26353/ide-%DA%86%DB%8C%D8%B3%D8%AA%D8%9F-%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-ide-%D8%A8%D9%87-%D8%B2%D8%A8%D8%A7%D9%86-%D8%B3%D8%A7%D8%AF%D9%87**](https://programming.tosinso.com/fa/articles/26353/ide-%DA%86%DB%8C%D8%B3%D8%AA%D8%9F-%D8%A2%D8%B4%D9%86%D8%A7%DB%8C%DB%8C-%D8%A8%D8%A7-ide-%D8%A8%D9%87-%D8%B2%D8%A8%D8%A7%D9%86-%D8%B3%D8%A7%D8%AF%D9%87)

امروزه بعید به نظر می رسد که کسی در دنیای برنامه نویسی دستی بر آتش داشته باشد اما اصطلاح API به گوش او نخورده باشد! اگر شما به عنوان یک برنامه نویس با ای پی آی ها کار کرده باشید از اهمیت آن ها مطلع هستید، با این حال سایر افراد ممکن است بپرسند که «**اصلاً ای پی آی ها چه هستند و چرا تا این حد مورد توجه قرار می گیرند؟**» در این آموزش به دنبال پاسخ دادن به این پرسش ها هستیم و می خواهیم ببینیم یک API خوش ساخت از چه ویژگی هایی برخوردار است.

اگر بخواهیم تعریف ساده ای از API که مخفف عبارت Application Programming Interface ی «واسط برنامه نویسی اپلیکیشن»است داشته باشیم باید بگوییم که ای پی آی ها مجموعه مقرراتی هستند که مشخص می کنند چه طور یک اپلیکیشن می تواند با اپلیکیشن دیگری ارتباط برقرار کند. زمانی که از یک پی سی یا لپ تاپ استفاده می کنیم، ای پی آی ها همان چیزهایی هستند که انتقال اطلاعات را بین برنامه ها امکان پذیر می کنند، برای مثال ای پی آی های سیستمی، اجرای برنامه ای مانند آفیس ورد را روی سیستم عامل ویندوز امکان پذیر می سازند.

همان طور که از نام API مشخص است، آن ها مانند یک رابط بین برنامه نویسان و یک اپلیکیشن هستند؛ به این صورت که اگر برنامه نویس برای پیش برد برنامه ی خود نیاز به اطلاعاتی از اپلیکیشن دیگری داشت، این واسط ها به او می گویند که می تواند برای دریافت چه اطلاعاتی درخواست دهد، چگونه درخواست خود را مطرح کند، و چگونه اطلاعات خود را دریافت کنند. در واقع API هر اپلیکیشن درخواست های مجاز را از سایر برنامه ها تحویل می گیرد، به اپلیکیشن مورد نظر می فرستد، آن گاه اطلاعات را دریافت می کند و تحویل برنامه های دیگر می دهد.

API ها در فضای وب هم برای سرویس هایی مانند گوگل مپ یا فیسبوک این امکان را فراهم می کنند تا این برنامه ها به اپلیکیشن های دیگر اجازه دهند که آن ها را اجرا کنند. برای مثال یک اپلیکیشن Yelp با استفاده از ای پی آی ها می تواند نزدیک ترین رستوران در اطراف کاربر را روی نقشه های گوگل مپ نشان دهد. یا یک اپ بازی می تواند با سرویس ابری دراپ باکس ارتباط برقرار کند تا تاریخچه ای از نتایج بازی های کاربر روی حافظه ی ابری این سرویس ذخیره شود، بدون آن که توسعه دهنده مجبور باشد یک ذخیره ساز ابری مخصوص اپلیکیشن خود طراحی کند.

اکنون باید ببینیم که ای پی آی ها چگونه این ارتباطات را امکان پذیر می کنند. در حقیقت ای پی آی ها با استفاده از روشی محدود شده و در چارچوب قوانینی خاص با "افشای" برخی از توابع داخلی یک برنامه به محیط خارج از آن تمام این کارها را انجام می دهند. با این کار بدون آن که نیاز باشد توسعه دهنده ها تمام سورس کدهای برنامه ی خود را در اختیار دیگران قرار دهند، اپلیکیشن ها می توانند با به اشتراک گذاری بخشی از داده ها با هم ارتباط برقرار کرده و تعامل داشته باشند.

API ها با محدود کردن دسترسی برنامه های خارجی به بخش خاصی از قابلیت های یک برنامه -که اغلب هم این سطح از دسترسی کافی است- تمام اطلاعات مورد نیاز سایر اپلیکیشن ها برای ارتباط با نرم افزار مورد نظر را فراهم می کنند. در حقیقت می توان API ها را هم چون پنجره ای به داخل یک برنامه تصور کرد.

با توجه به تمام مثال هایی که از کاربرد API های مختلف بیان کردیم، به نظر می رسد که متوجه اهمیت آن ها در حوزه ی توسعه ی نرم افزار شده باشیم. همان طور که اعضای یک جامعه برای زندگی در کنار هم به یکدیگر نیاز دارند و باید با هم در تعامل باشند، هر چند که در بسیاری از موارد نیازی نیست یک شناخت عمیق از هم داشته باشند -مثلاً برای خرید سبزی، شما لازم نیست که در مورد زندگی خصوصی سبزی فروش محلتان اطلاعات داشته باشید، کافی است که بدانید چه کسی سبزی فروش است و مغازه ی او کجا است- این روزها اپلیکیشن های مختلف نیز به شدت نیاز به تعامل با هم دارند و برای این کار لازم است اطلاعاتی را در مورد هم داشته باشند. ای پی آی ها که تأمین کننده ی این اطلاعات هستند، علاوه بر حفظ امنیت اپلیکیشن ها، با فراهم کردن اطلاعات لازم و حذف داده های غیر ضروری، دقت و سرعت توسعه ی نرم افزارها را بالا می برند. حال باید ببینیم خصوصیات یک API خوب کدامند؟ می توان گفت که یک ای پی آی خوب، ای پی آی یی است که:

 - یادگیری آن آسان باشد؛  
- کاربرد آن حتی بدون استفاده از اسناد، باز هم آسان باشد؛  
- امکان استفاده ی نادرست از آن کم باشد؛  
- خوانایی و امکان نگهداری و پشتیبانی از کدهایی که از آن استفاده می کنند بالا باشد؛  
- برای رفع نیازهای ارتباطی نرم افزارهای دیگر با آن به اندازه ی کافی قدرتمند باشد؛  
- توسعه و گسترش آن ساده باشد؛

برای این که چنین خصوصیاتی را در یک ای پی آی ایجاد کنیم لازم است در زمان طراحی آن قواعدی را رعایت کنیم. از جمله این قواعد می توان به موارد زیر اشاره کرد:

- هر API یک وظیفه دارد و باید آن را به بهترین نحو ممکن انجام دهد. کارکرد یک API باید به گونه ای باشد که بتوان آن را به ساده ترین صورت ممکن بیان کرد. به طور مثال می گوییم وظیفه ی یک ای پی آی این است که امکان استفاده از شبکه ی اجتماعی یوتیوب را در داخل اپلیکیشن ها فراهم می کند. اگر نتوانید وظیفه ی یک API را به راحتی شرح دهید به این معنا است که مشکلی وجود دارد، و برای رفع آن می توانید طراحی ای پی آی را در قالب ماژول های کوچک تر پیاده سازی کنید، که هر کدام وظیفه ی خاصی داشته باشند.

- باید API تا حد ممکن کوچک باشند؛ هر چه اطلاعات کمتری افشا شود بهتر است. با این حال باید دقت داشته باشیم که ای پی آی طراحی شده حاوی تمام اطلاعات ضروری باشد. تنها به خاطر داشته باشید که همیشه امکان اضافه کردن اطلاعات وجود دارد، اما حذف اطلاعات افشا شده غیر ممکن است!

- این API ها نباید از پیاده سازی های مختلف تأثیر بگیرند. به عبارت دیگر، روش استفاده ی واحدی از آن API وجود داشته باشد.

- دسترسی ها به هر چیز را در پایین ترین سطح ممکن قرار دهید. کلاس ها و آبجکت ها را تا جای ممکن private کنید. کلاس های public نباید فیلدهای public داشته باشند (به استثنای ثابت ها).

- در زمان طراحی و ساخت API سعی کنید که از کاربران نهایی خود فیدبک بگیرید تا مطمئن شوید محصول نهایی شما تمام نیازهای معقول آن ها را برآورده می کند. به علاوه، باید بدانید که طراحی ای پی آی ها نیاز به یک کار تیمی دارد و به شدت پیشنهاد می شود که تمام اعضای تیم طراحی نرم افزار روی طراحی ای پی آی هم کار کنند.

- در طراحی API از تکرارهای بیش از حد اجتناب کنید. برای مثال اگر دو متد یک کار را انجام می دهند، نباید از هر دوی آن ها استفاده کنیم. این کار باعث سردرگمی کاربران می شود.

- از پروتکل امن SSL استفاده کنید. بیش تر کاربران از نقاط دسترسی غیر رمز گذاری شده برای دسترسی به اینترنت استفاده می کنند که امکان هک شدن آن ها را به راحتی فراهم می کند.

نکته

SSL مخفف واژگان Secure Sockets Layer به معنی «لایه ی سوکت های ایمن» یکی از پروتوکل های رمزنگاری استاندارد است که به منظور تأمین امنیت ارتباطات اینترنتی توسط شرکت Netscape به منظور انتقال داده‌های خصوصی از طریق اینترنت توسعه داده شد. این پروتوکل امنیت انتقال داده‌ها را در اینترنت برای مقاصدی همچون درگاه های بانک، تصدیق اطلاعات کاربری، انتقال اطلاعات هویتی و دیگر داده‌های حساس امکان پذیر می سازد. به طور خلاصه، کارکرد پروتوکل اس اس ال به این شکل است که مابین کاربر و سایت -که به طور معمول نماینده ی کاربر مرورگری است همچون فایرفاکس یا گوگل کروم و نماینده ی سایت هم وب سرور است- یک لینک رمزنگاری شده ایجاد می‌گردد. در شرایط عادی، اطلاعاتی از این دست به صورت Plain Text یا «متن ساده» مابین مرورگرها و وب سرورها رد و بدل می‌شوند که بسیار آسیب‌پذیر است اما در اس اس ال، کلیه ی داده های رمزنگاری می شوند و به طور کلی، یو آر ال هایی که نیاز به ارتباط اس اس ال دارند، به جای http، با https شروع می شوند.

- از نوشتن مستندات غفلت نکنید. این موضوعی است که بسیاری از برنامه نویسان آن را نادیده می گیرند یا به درستی انجام نمی دهند. مستندات خود را به صورت دقیق و ساده تهیه کنید. بهترین تاکتیک برای این کار این است که فکر کنید می خواهید موضوع را برای یک کودک 10 ساله شرح دهید. از نوشتن در مورد هیچ مرحله ای اجتناب نکنید و هرگز این پیش فرض را نداشته باشید که چون موضوعی برای شما بدیهی است دیگران هم باید آن را بدانند. مستندسازی مناسب به ویژه در شرایطی که فرآیندها پیچیده اند کمک بزرگی به کاربران خواهد کرد.

- واقع گرا باشید. به هر حال با وجود رعایت تمامی قواعد و مشورت اعضای تیم با یکدیگر باز هم امکان اشتباه و نادیده گرفتن یک سری مسائل وجود دارد. اما آن چه باید بدانید این است که در نهایت شما نمی توانید رضایت همه ی افراد را به دست آورید.

در انتها باید این نکته را به شما یادآوری کنیم که هر چند API ها ابزارهایی سودمند در دست برنامه نویسان هستند اما استفاده از آن ها با چالش هایی همراه است. باید بدانید که اگر یک API امروز در دسترس کاربران قرار دارد دلیلی ندارد که این دسترسی برای همیشه وجود داشته باشد. برای مثال چند سال پیش شرکت توئیتر دسترسی کاربران را به ای پی آی های خود محدود کرد.

علاوه بر این، کمپانی هایی که API های خود را در دسترس کاربران قرار می دهند ممکن است بعد از مدتی به ارائه ی خدمات خود پایان دهند. در این صورت اگر اپلیکیشن شما وابسته به API های این شرکت ها باشند با مشکل مواجه خواهند شد. برای مثال فرض کنید شرکت گوگل تصمیم بگیرد ارائه ی خدمات سرویس گوگل مپ را متوقف کند، در این صورت کار اپلیکیشن Yelp نیز متوقف خواهد شد. به هر حال با وجود تمام چالش هایی از این دست، توسعه دهندگان علاقه ی زیادی به استفاده از API ها دارند و هر روز شاهد گسترش ای پی آی های مختلف هستیم (در صورت تمایل به کسب اطلاعات بیشتر در مورد ای پی آی ها، می توانید به فصل پنجم از دوره ی [آموزش اصول برنامه‌نویسی](https://sokanacademy.com/courses/coding) در سکان آکادمی نیز مراجعه نمایید.)