■ **یک پروژه موفق متن‌باز در گیت‌هاب از شش جز اساسی تشکیل شده است.**

جز اول – اسناد

مدارک یا اسناد در گیت‌هاب به‌صورت کد هستند. «توسعه‌دهنده شخص ثالث (Third-party developer) «نحوه استفاده از این روش ارائه متن را درک می کند. بنابراین، شانس او برای گرفتن امتیاز بین همکاران و دنبال کنندگانش بیشتر می‌شود. در حال حاضر GitHub دستورالعمل‌هایی را در مورد نحوه تهیه اسناد، تدوین کرده است.

جز دوم – کدها پاسخگوی نیاز عموم باشند

کدها لازم است مجدداً قابل استفاده باشند و مشکلی را رفع کنند که در بین عموم رایج است. برای مثال، تصور کنید در طی توسعه‌ی یک اپلیکیشن اندرویدی در مبحثی خاص به مشکل برخورده‌اید، این مشکل عمومیت دارد و البته کسی نیز راه حل اوپن‌سورس و رایگانی را برای آن فراهم نکرده است. اگر شما برای مسئله‌ی مذکور راه‌حلی پیاده‌سازی کنید و آن را روی گیت‌هاب در قالب یک کتابخانه در اختیار عموم بگذارید، به احتمال زیاد کتابخانه‌ی شما بسیار سریع توسط سایر توسعه‌دهندگان شناخته شده، به محبوبیت رسیده و مرتباً در اپلیکیشن‌های گوناگون مورد بهره‌برداری قرار خواهد گرفت.

جز سوم – تعداد باگ‌ها

کد بی‌عیب و نقص، رویای مشتریان و کابوس یک متخصص QC است. در زندگی واقعی، کد ۱۰۰ درصد کاملی وجود ندارد. اگر تعداد باگ‌های کد نوشته‌شده، کم باشد به احتمال زیاد مخاطب مناسب خود را به دست خواهید آورد. شرکت‌های بزرگی که کاربر گیت‌هاب هستند از اینکه یک برنامه‌نویس بتواند در محصولشان اشکالی پیدا کند بسیار استقبال می‌کنند. شرکت‌هایی مانند گوگل (Google) فیس‌بوک (Facebook) و موزیلا (Mozilla)برای کسانی که باگ‌هایشان را شناسایی کنند جایزه نقدی در نظر گرفته‌اند.

معرفی متخصص :QC کارشناس کنترل کیفیت (Quality Control, QC) که جزئی از مدیریت کیفیت است، بر برآورده نمودن و تکمیل پروتکل‌های کیفیت، تمرکز و توجه دارد. کار اصلی یک متخصص QC اطمینان یافتن از کیفیت استاندارد یک خروجی است.

مفهوم باگ :باگ (Bug) در لغت به معنای حشره است. در برنامه‌نویسی به خطاهای کوچکی که در برنامه به وجود می‌آید، باگ می‌گویند. برای نخستین بار در سال ۱۹۴۵ در دانشگاه هاروارد (Harvard University) خانمی به نام گریس هاپر (Grace Hopper) نخستین مادر و مدرس برنامه‌نویسی، زمانی که در حال عیب‌یابی یک ماشین‌حساب به نام Mark II بود لغت باگ را به‌کاربرد. گفته می‌شود که در آن زمان، ماشین‌حساب از کارافتاده بود و خانم هاپر بعد از مدتی طولانی جست‌وجو متوجه می‌شود که یک سوسک بین قطعه‌های الکترونیکی ماشین‌حساب، گیر کرده است! و این گونه «باگ» وارد دنیای برنامه‌نویسی شد. به فرایند پیدا کردن و حذف ایرادها از برنامه، دیباگ (Debug) گفته می‌شود.

جز چهارم – کیفیت یک کد

کیفیت یک کد با پارامترهای زیر مشخص می‌شود:

- **پارامتر اول** - کد باید به‌گونه‌ای نوشته شود که نگهداری و گسترش آن در آینده، کار آسانی باشد.
- **پارامتر دوم** - کد باید با استانداردهای پذیرفته شده در زبان برنامه‌نویسی مطابقت داشته باشد.
- **پارامتر سوم** - کد باید قابل خواندن و ساختار یافته باشد. توصیه متخصصان، استفاده از الگوهای طراحی موجود است. زیرا این الگوها راه‌حل‌هایی را برای مواجه با موقعیت‌های تکراری توسعه‌دهندگان ارائه می‌دهند. اگر برای کدهایی که در حال نوشتن آن هستید کتابخانه‌های شخص ثالثی وجود دارند، به‌جای آزمون و خطا از آنها استفاده کنید.
- **پارامتر چهارم** - کد با کیفیت کدی است که آزموده شده باشد یا به زبان دیگر قابل اعتماد باشد. هر چند عملکرد «رابط کاربری (Interface)»را می‌توان به صورت دستی و انسانی کنترل کرد، ولی کدهای خام به صورت معمول توسط «تست‌های واحدی (Unit Tests)»پوشش داده می‌شوند (کدهای خام مانند سایر کدهای معمول به نظر می‌رسند، ولی برنامه‌نویسان از طریق آنها بخش به بخش عملکرد اجزای برنامه خود را می‌سنجند). در حالت ایده‌آل اگر بتوانیم ۱۰۰ درصد کدهای خام خود را تحت پوشش آزمون‌های واحدی قرار دهیم، به یقین از هر باگ غیر منتظره‌ای در امان خواهیم بود. اما اگر انجام این حجم از تست هزینه‌های ما را سرسام‌آور می‌کند، شاید بهتر باشد حداقل بخش‌های کلیدی و حیاتی کد خود را مشخص کرده و آنها را مورد آزمون قرار دهیم.
- **پارامتر پنجم** - هنگام نوشتن کد، اصول DRY و KISS را به یاد داشته باشید. آنها به بهبود کد شما کمک خواهند کرد.
  - **اصل DRY**: خودت را تکرار نکن (Don't Repeat Yourself, DRY) اصلی در برنامه‌نویسی رایانه‌ای است که به‌وسیله عدم تکرار یک یا چند خط کد در مکان‌های مختلف برنامه، رعایت می‌شود. درواقع با این کار برای اصلاح قسمتی از برنامه، نیازی به اصلاح قسمت‌های مختلف کد وجود ندارد. این اصل می‌گوید که هر بخشی از برنامه باید یک نمود یکتا، نامبهم و معتبر داشته باشد. برنامه‌نویسان برای رعایت این اصل، کد خود را داخل یک تابع یا کلاس قرار می‌دهند و در موارد موردنیاز تابع را فراخوانی کرده یا یک شی جدید از کلاس می‌سازند. بر طبق این اصل، هر برنامه‌نویس دقایقی پس از نوشتن چند خط اول، مراحل نگهداری یا پشتیبانی از کد خود را آغاز می‌کند.

- اصل **KISS:** آن را ساده و احمق نگه‌دار (Keep It Simple, Stupid, KISS) اصلی است که توسط کلی جانسون (Kelly Johnson)معرفی شد. این اصل بیان می‌کند که اگر سیستم‌ها به دنبال ساده‌ترین و بدیهی‌ترین روش‌ها برای انجام کارها باشند عملکرد بهتری خواهند داشت. بنابراین، سادگی باید هدف اصلی طراحی سیستم‌ها باشد و از پیچیدگی‌های بیهوده اجتناب کرد.

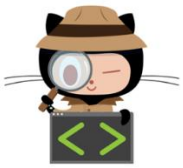
- **ترویس و یکپارچه‌سازی کدها**

بسیاری از تست‌های کیفیت کد را می‌توان اتوماسیون کرده تا به صورت خودکار انجام شوند. ناگفته نماند که کتابخانه‌هایی نیز برای تست‌های واحدی وجود دارند، که نوشتن و پیاده‌سازی آنها را به مراتب ساده‌تر می‌سازند. لازم است اشاره کنیم که پروژه‌های اوپن سورس سنجش کیفیت با سرورهای CIپیکربندی می‌شوند و وظیفه‌شان این است که تمامی چک‌ها یا کنترل‌های تعریف شده را در حین هر درخواست «Pull» به صورت خودکار اجرا کنند. حال می‌بایست به این نکته بپردازیم که چنانچه یکی از درخواست‌های «Pull» با پاسخ صحیحی مواجه نشود کاربری که آن را ارسال کرده می‌بایست علت آن را بررسی کرده و پس از یافتن مشکل نسبت به رفعش اقدام نماید. بد نیست بدانیدTravis CIمحبوب‌ترین سرور CI محسوب می‌شود و می‌توانید آن را به رایگان به ریپازیتوری عمومی خود متصل کنید.

جز پنجم - پشتیبانی فعال

اگر یک راه‌حل جدید ارسال می‌کنید باید خود را برای دریافت پاسخ‌های دیگران آماده‌سازید. شاید برخی از کاربران، مشکل‌های موجود را بیان کنند، تغییر جدیدی را پیشنهاد دهند یا فقط پیگیر پروژه باشند. یک قانون خوب در گیت‌هاب این است که شما باید به فعالیت‌های کاربران، پاسخ دهید. حمایت از پروژه، تصحیح سریع خطاها و البته باز بودن پروژه برای تغییر، کلید موفقیت شما است. توسعه نسخه‌های جدید یک پروژه، مزیت بزرگی در بانک اطلاعاتی شما به شمار می‌رود.

جز ششم- مجوز یا سطح دسترسی در گیت‌هاب



نرم‌افزارهای متن‌باز، نرم‌افزارهای رایگانی هستند. همه برنامه‌ها و موارد موجود در Github را با یک اخطار کوچک و تا جایی که مجوزهای برنامه اجازه می‌دهند، می‌توان به‌صورت رایگان استفاده کرد. به دلیل تمایل زیادی که برای به اشتراک‌گذاری و باز کردن فایل‌های متن‌باز در جهان وجود دارد، گیت‌هاب به کاربرانش امکان انتخاب یک نوع مجوز با دسترسی خاصی را می‌دهد. به‌عنوان‌مثال

- **مجوز Unlicense :** مجوزی بدون شرط و بر اساس متن‌باز است. این مجوز زمانی صادر می‌شود که کد مالکیت، عمومی باشد. انجام چندین پروژه عملیاتی، اصلاح و افزودن، برای همه کاربران بدون استثنا در دسترس خواهد بود.
- **مجوز MIT :** کاربران حق استفاده آزادانه از این کد را دارند و با آن قادر به ارائه پیشنهادهای خود هستند. بااین‌حال، کاربران باید حقوق کپی‌رایت را رعایت کنند. چنین مجوزی از jQuery ، NET.، Core و Rails استفاده می‌کند.
- **مجوز Apache 2.0 :** علاوه بر MIT ، مجوز ارائه اظهارنامه حق ثبت اختراع را به کاربرانش می‌دهد. این مجوز توسط اندروید(Android) ، آپاچی (Apache) و سویفت (Swift) استفاده می‌شود.
- **مجوزهای اختصاصی بسته :** در گیت‌هاب، پروژه‌هایی با یک مجوز بسته وجود دارند. در پروژه‌هایی با این مجوز، فقط یک شرکت‌کننده خاص، امکان دسترسی به داده‌ها را دارد.

صدرنشینان رتبه‌بندی مخازن گیت‌هاب

در رتبه‌بندی فعلی مخازن گیت‌هاب که برحسب زبان برنامه‌نویسی و تعداد ستاره‌های دریافتی انجام‌گرفته است به ترتیب جاوااسکریپت (JavaScript)سپس پی‌اچ‌پی (PHP) و در انتها جاوا (Java) عنوان‌های اول تا سوم این رتبه‌بندی را از آن خود کرده‌اند. به یاد داشته باشید در دنیای بی‌انتهای برنامه‌نویسی، گوی قدرت در دست کسانی است که بی‌توجه به اصول و قانون‌های دست‌وپا گیر، در فکر پیدا کردن راهی بهتر، سریع‌تر و ساده‌تر هستند. کسانی که در آینده از آنها به بزرگی یاد خواهد شد.

آشنایی با فایل README و سینتکس Markdown

فایل README

چرا هر پروژه‌ای به یک فایل README خوب نیاز دارد؟

فایل README را می‌توان یک راهنما دانست که توضیحات مفصلی از پروژه‌ی قرار گرفته شده در ریپازیتوری GitHub را ارائه می‌دهد در واقع

فایل README فایل توضیحات اصلی پروژه شماست! هرکسی که وارد صفحه پروژه میشه پایین تر از فایل ها این توضیحات رو میبیندو اگر فایل README رو درست ننوشته باشید یا اصلا ننوشته باشید خیلی از کسایی که پروژه شما رو میبینند متوجه کارایی پروژه نمیشوند! و دلایل دیگر عبارتند از:

- یک فایل README خوب می‌تواند پروژه‌ی شما را از دیگر پروژه‌های متن باز متمایز کند و به‌همین منظور باید فایل README به‌خوبی خود پروژه باشد.
- این اولین فایلی است که اشخاص هنگام مواجه شدن با پروژه‌ی شما مشاهده می‌کنند بنابراین باید قادر باشید پروژه‌ی خود را به‌طور مختصر اما جزئی شرح دهید.
- می‌توانید به‌کمک این فایل بر روی چگونگی ارائه‌ی پروژه‌ی خود تمرکز کنید.

- هنگام نوشتن فایل README درنظر داشته باشید که سایر توسعه‌دهندگان می‌خواهند به‌کمک این فایل، پروژه‌ی شما را درک کنند.

■ ویژگی‌های یک فایل README خوب

قبل از شروع نوشتن فایل README باید سوال‌های زیر را از خود بپرسید:

- انگیزه‌ی شما از توسعه‌ی این پروژه چه بوده است؟
- این پروژه چه مشکلی را حل می‌کند؟
- مزایا پروژه‌ی شما نسبت به سایر پروژه‌ها چیست؟ اگر پروژه‌های شما ویژگی‌های زیادی دارد می‌توانید یک بخش با عنوان Features به فایل README اضافه کنید.

پس از پاسخ به این سوال‌ها باید جزئیاتی را در نوشتن فایل README رعایت کنید.

□ عنوان پروژه

شما می‌توانید نام پروژه را در عنوان فایل README وارد کنید که توصیف کننده‌ی کل پروژه است و به سایر افراد در درک هدف اصلی این پروژه کمک می‌کند.

□ توضیحات

بخش توضیحات را می‌توان جنبه‌ی بسیار مهمی از فایل README دانست که به‌کمک آن می‌توانید پروژه‌ی خود را به سایر توسعه‌دهندگان یا حتی کارفرهای بالقوه ارائه دهید. همچنین توجه داشته باشید که کیفیت توضیحات می‌تواند پروژه‌ی شما را از یک پروژه‌ی بد، متمایز کند.

- برنامه‌ی شما چه کاری انجام می‌دهد؟
- دلیل شما برای استفاده از فناوری‌های فعلی چه بوده است؟
- با چه چالش‌های روبرو شده‌اید و چه ویژگی‌های امیدوار کننده‌ای در آینده به این پروژه اضافه می‌شود؟

□ فهرست مطالب (اختیاری)

اگر فایل README شما بسیار طولانی باشد، بهتر است فهرست مطالب را به فایل README پروژه‌ی خود اضافه کنید تا کاربران راحت‌تر به آنچه که نیاز دارند، دسترسی پیدا کنند.

□ نحوه‌ی نصب و پیکربندی پروژه

اگر پروژه‌ی شما یک نرم‌افزار یا برنامه‌ای است که نیاز به نصب یا پیکربندی داشته باشد باید مراحل لازم برای نصب و پیکربندی پروژه را در فایل READMEقرار دهید.

□ چگونگی استفاده از پروژه

دستورالعمل‌هایی ارائه دهید که سایر توسعه‌دهندگان یا حتی کاربران عادی قادر باشند از پروژه‌ی شما استفاده کنند. همچنین می‌توانید از اسکرین شات وتصاویری برای نشان دادن پروژه‌های در حال اجرا استفاده کنید..از ساختار و اصول طراحی مورد استفاده در پروژه خود یاکمک های بصری استفاده کنید اگر پروژه شما نیاز به احراز هویت مانند گذرواژه یا نام کاربری دارد، این بخش خوبی برای گنجاندن موارد است.

اعتبارات شامل

اگر به عنوان یک تیم یا سازمان روی پروژه کار کرده اید، همکاران/اعضای تیم خود را فهرست کنید. همچنین باید پیوندهایی به پروفایل های GitHub و رسانه های اجتماعی آنها نیز اضافه کنید.و اگر آموزش‌ها را دنبال کرده‌اید یا به مطالب خاصی اشاره کرده‌اید که ممکن است به کاربر در ساخت آن پروژه کمک کند، پیوندهایی به آن‌ها در اینجا نیز وارد کنید.

لایسنس (مجوز ها)

اضافه کردن لایسنس به پروژه باعث می‌شود تا سایر توسعه‌دهندگان بدانند چه کارهایی را می‌توانند با پروژه‌ی شما انجام دهندو و چه کاری نمی توانند انجام دهند در اکثر فایل های README، این معمولا آخرین قسمت در نظر گرفته می شود

بسته به نوع پروژه ای که روی آن کار می کنید، و یا موردی که انتخاب می‌کنید، سهمی که پروژه برای شما تعیین می‌کند. انواع مختلفی از مجوزها را داریم می توانید از طریق این سایت به انواع مجوزها دسترسی داشته باشید https://choosealicense.com/

badgeها

اضافه کردنbadge ها ضروری نیست اما به‌کمک آن‌ها می‌توانید برخی جزئیات کلیدی را به ساده‌ترین شکل ممکن به فایل README اضافه کنید.

داشتن این بخش همچنین می تواند برای کمک به پیوند به ابزارهای مهم مفید باشد و همچنین برخی از آمارهای ساده در مورد پروژه شما مانند تعداد فورک ها، مشارکت کنندگان، مسائل باز و غیره را نشان می دهد. .در زیر یک اسکرین شات از یکی از پروژه های است که نشان می دهد چگونه می توانید از نشان ها استفاده کنید:

خوبی این بخش این است که به طور خودکار خود به روز می شود.

این نشان ها را می توانید از نشان های میزبانی شده توسط shields.io جستجو وتهیه کنید. آنها تعداد زیادی نشان برای کمک به شما در شروع کار دارند. شما شاید در ابتدا متوجه نشوید که همه آنها اکنون چه چیزی را نشان می دهند، اما به مرور متوجه خواهید شد.

نحوه‌ی مشارکت در پروژه

اگر برنامه یا پکیجی ایجاد کرده‌اید و می‌خواهید سایر توسعه‌دهندگان در توسعه‌ی پروژه‌ی متن باز شما مشارکت داشته باشند باید دستورالعمل‌های مشخصی را در فایل README قرار دهید.

تست ها

برای درخواست خود تست بنویسید. سپس نمونه کد و نحوه اجرای آنها را ارائه دهید.این به شما کمک می کند نشان دهید که مطمئن هستید که پروژه شما بدون هیچ چالشی کار می کندو این کار باعث می‌شود افراد دیگر نیز به آن اعتماد کنند.

امتیاز اضافی

- آن را به روز نگه دارید - یک روش خوبی است که مطمئن شوید فایل شما همیشه به روز است. در صورت وجود تغییرات، حتما فایل را در صورت لزوم به روز کنید.
- یک زبان را انتخاب کنید - همه ما از مناطق مختلف جهان و با زبان های مختلفی صحبت می کنیم. اما این بدان معنا نیست که باید کد خود را به زبان محلی ترجمه کنید. نوشتن README به زبان انگلیسی بهتر زبان خواهد بود زیرا انگلیسی یک زبان پذیرفته شده جهانی است. اگر مخاطب هدف شما با زبان انگلیسی آشنا نیست، ممکن است بخواهید در اینجا از ابزار مترجم استفاده کنید.

بسته بندی کنید

همه چیزهایی که برای بهبود فایل های README یا حتی شروع نوشتن اولین فایل خود به آن نیاز دارید، وجود دارد.در این مرحله من مطمئن هستم که شما در موقعیتی هستید که می توانید یک راهنمای تعاملی و اطلاعاتی را به پروژه بعدی خود یا حتی پروژه موجود اضافه کنید و پروژه خود را برجسته کنید.ابزارهای زیادی وجود دارد که می توانید از آنها برای ایجاد خودکار README برای پروژه خود نیز استفاده کنید، اما همیشه روش خوب این است که قبل از حرکت به سمت اتوماسیون آن را به تنهایی امتحان کنید

- Make a README
- README Generator
- README

https://www.freecodecamp.org/news/how-to-write-a-good-readme-file/

https://rahuldkjain.github.io/gh-profile-readme-generator/

https://github.com/kefranabg/readme-md-generator

نام فایل هم باید حتما README.md باشد