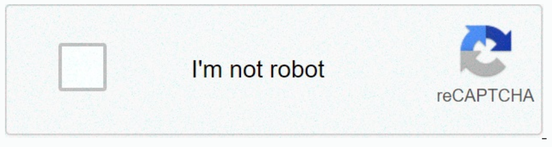

Continue

Story saver apk app

Among the programs I always download when first setting up a new computer is FileZilla, the humble, open source FTP client. For those of us who have run our own small websites it has been a long time staple in the webmaster toolbox. Although I am so purely functional, I can't help but feel nostalgic when using FileZilla. It reminds me of being a child and trying to encode HTML so that I could put cool pictures and GIFs on my very own website. And once the code was fixed, it was time to fire up the FTP software and upload to the server, the last step in publishing. Nowadays most sites can be fully designed and maintained within a web browser, but I still keep FileZilla on hand to tinker with my own projects. How did FileZilla come about? Like a school project, actually. We spoke to developer Tim Kosse about how and why he created FileZilla about 15 years ago. So FileZilla actually started out as a school project, is that right? Were you trying to solve a problem you had experienced with existing software, or did the inspiration come from somewhere else? Yes, it started as a group project in the computer science course during my senior year of school, almost 15 years ago. We were looking for something useful that was neither too trivial nor too big to implement and could be worked on in parallel. Unfortunately, I no longer remember the details of how we eventually settled on an FTP client. G/O Media can get a commissionAfter you came up with the idea, what was the next step? Figure out how to implement the most basic set of features every FTP client needs: Connect to a server, get directory listings and of course upload and download of files. With GUI design we did imitate the basic layout of other graphic FTP clients that are known to us. For the backed functionality, we used some third-party FTP classes. After the project's official conclusion I continued to work on it, partly because I was able to make use of a good FTP client myself. Gradually I added features and fixed the bugs when they appeared. At some point I also started with FileZilla Server to supplement the client. How did you choose which platforms to target and which ones to ignore or wait for? Originally FileZilla was for Windows only, it was the only platform we had enough experience with when we started the project. At some point though I wanted to expand on other platforms and Linux in particular. Unfortunately, due to the original choice of using Microsoft's MFC as a GUI library, supporting other platforms would have meant a complete rewrite. Early in 2004, I finally decided to write about FileZilla from scratch, [as] the old code base had become increasingly difficult to maintain and adding new features had a great chance of breaking things. After some research, I eventually settled on the wxWidgets library and started writing about in 2014, which culminated in filezilla 3.0.0 release just three and a half years later. Not only did I seriously underestimate the complexity of cross-platform cross-platform during all this time I still had to keep the old version until the parity feature was reached. Furthermore, I kept improving FileZilla Server during this time. Yet the euphemism has been a success. What was your biggest roadblock and how did you overcome it? It's more like an endless collection of unnecessary speed bumps that have hampered me the most. FTP is a pretty old protocol with lots of history and older functionality. Parts of FTP are specified in detail while other parts are left unspecified. Not only are there often different ways to achieve the same thing in FTP, different FTP software supports and uses different features. In addition, FTP's use of separate connections for control commands and data transfers makes it difficult to use behind firewalls and NAT routers, which is an additional challenge in itself. This leads to interesting problems: For example, a server may not implement a feature correctly or a firewall in front of the server may choke on some commands. Furthermore, FileZilla may be the only client that uses this command. As a result, FileZilla cannot be used to exchange files with such a server. Something like this happens almost every time I use an additional FTP feature or change how FileZilla works. In these situations I spend a lot of time and effort to determine the root cause of the problem. When this happens in other open source FTP software I usually write a patch that fixes the issue. Sadly, it often happens with proprietary servers, firewalls and NAT routers where I can't do anything myself to fix it, so the respective vendors have to solve the problem. It is at this exact point where my biggest roadblock lies: How to convince others that the problem is in a third-party component, especially in light of uninformed counterarguments that it works with other FTP clients or in an earlier version of FileZilla? Although I was able to implement solutions in such cases, they often come with a heavy cost and lots of extra complexity. Furthermore, the underlying problem would remain unfixed. Thus, I made an important decision: I will not implement any solutions if it would compromise security or performance. While FileZilla does not require a server to support optional features, if the server says it supports a particular feature then FileZilla in turn expects the server to handle it correctly. Sure, sometimes this can be most inconvenient for affected users, especially if the server or firewall provider does not recognize or do not want to fix the problem. Nevertheless, I am convinced that solving the root cause of a problem is the right thing to do. People have a lot of options when choosing FTP software; what features were important to you when designing FileZilla? Before deciding whether to implement a new feature in FileZilla I look at whether a graphic FTP client is the right tool for the right task. It some value in implementing a function that would be better suited to other types of One such example would be automation and script transfers, I think this would be a much better fit for a simple command line client. After deciding to implement a new feature, there are two primary design considerations guiding the implementation of the new feature in FileZilla.The first is reliability. Each feature must work as advertised in all possible usage scenarios. In addition, I consider security one aspect of reliability as well: A security risk can be seen as software that does more than it is intended to do. In other words, users can trust FileZilla to do exactly what it is meant to do, not more or less. The other aspects are performance and scalability. Of course FileZilla should be as fast as possible, no one likes to wait longer than necessary for a task. When implementing a feature I always consider what happens if someone were to use it to the utmost. For example, most users probably only deal with a few thousand files, yet FileZilla has been designed to handle directories containing millions of files totaling multiple terabytes. After it lived beyond its academic purpose, what was launch that for you? I'm guessing it's been a bit of a slow snow ball, with more people adopting the software over time. Yes, the first edition would have been very quiet. I registered a new project on SourceForge, set up the project description, uploaded the first release and called it a day. Eventually the first users tried out FileZilla and sent valuable feedback. I kept improving FileZilla and in turn the user base continued to grow with each release. I never expected this much success, but it keeps me motivated to continue with the project. Had someone told me in 2001 that I would have millions of users, I would have called him crazy. Fun fact: The early versions from 2001 still work on a modern Windows 10 machine. Was there ever any thoughts of trying to sell it or make money from it somehow? I have never considered selling FileZilla as a viable option. After all, with FileZilla's free open source, redistribution is not only allowed, it is encouraged. But I participate in SourceForge's DevShare partnership program to generate revenue. I understand that some users object to bundled offers, so I have given all users freedom of choice, the use of the included installer is completely optional. How do you handle user requests and critiques effectively? User feedback can be broadly classified into three categories: Problem reports, feature requests, and general requests for help. Problems, I report triage immediately. If it turns out to be a bug in FileZilla it will likely be fixed in the next release. I think it's important to maintain a coherent product vision, so feature requests different ways. Many features would go ortatonal to the product vision, implementing them would quickly move FileZilla out of focus. This already filters out many feature requests. The rest may be prioritized and archived in the request tracker so that they can be implemented at some point. As for generic help requests, while I don't have time to respond to each such request myself, filezilla forums are a great site for users to help each other. Now, how do you split time between developing new features and managing existing ones? I always aim to have a bit of both in every new release. On average, I would say that two-thirds of my development time is spent implementing new features. Lately, however, I've spent a lot of time on architectural improvements, modernizing and cleaning up much of the codebase. While FileZilla still looks the same as it always did, much has changed under the hood. Not only has this greatly improved filezilla performance, it should also make it easier for new features to be implemented in the future. What advice would you give to others who want to take on a similar project? Be persistent and maintain a sharp product vision. Drop early and drop frequently to find out which of your focus areas needs to be improved or extended. Last but not least, use for yourself what you have created. Behind the App gives an insight into how some of our favorite apps came to be-from idea to launch (and beyond). Does anyone you want to see? E-mail Andy. Andy.