Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvantitativních metod

Nástroj GitHub Classroom a možnosti jeho využití

Bakalářská práce

Autor: Oleksandra Naboichenko
Studijní obor: Aplikovaná Informatika (ai3-p)

Vedoucí práce: doc. Ing. Filip Malý, Ph.D.

Hradec Králové březen 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 30.4.2021 Oleksandra Naboichenko

Poděkování:

Děkuji vedoucímu bakalářské práce, Ing. Filipu Malému, Ph.D za metodické vedení práce, za pomoc a rady při zpracování této práce.

Anotace

Tato diplomová práce byla zaměřena na podrobnou analýzu technologie GitHub a GitHub Classroom a její praktické aplikace ve vzdělávacím procesu na Univerzitě v Hradci Králové. Nejprve jsme se podívali na celý princip systému pro správu verzí a jejich varianty, a to lokální, centralizovaní a distribuovaní. Zkoumaly jsme již provedené studie o implementaci této technologie v jiných vzdělávacích institucích, abychom mohli nezávisle ocenit všechny pozitivní a negativní aspekty. Poté jsme rozebrali základní principy GitHub, a pak jsme se podrobně podívali na rozhraní GitHub Classroom a jeho schopnosti pracovat s perspektivou učitele a studenta.

Hlavním cílem práce bylo provést rozbor procesů práce jak se samotným GitHub, tak s jeho nástrojem GitHub Classroom pro vzdělávací účely. Tím poskytuji pro vzdělávací instituci podrobného přehledu a posloupnost práce s tímto serverem. Na závěr byl vypracován dotazník, který byl zaměřen na technickou stranu vzdělávacích institucí. Díky kterým bylo možné analyzovat vzdělávací systém v současné době a poskytnout Univerzitě Hradec Králové možnou alternativu pro zavádění nových technologií.

Annotation

Title: Github Classroom and its applications

This thesis was aimed at a detailed analysis of GitHub and GitHub Classroom technology, and its practical application in the educational process. First of all, we was looking at the whole principle of the version control system and their varieties, namely local, centralized and distributed. We have studied the already conducted research on the implementation of this technology in other educational institutions in order to independently assess all the positive and negative aspects. After that, we analyzed the basic principles of GitHub, after which we examined in detail the GitHub Classroom interface and its capabilities from the perspective of a teacher and a student.

The main purpose of the work was to analyze the process of working both with GitHub itself and with GitHub Classroom for educational purposes. By providing the school with a detailed overview and the sequence of work with the

server. As a completion, was worked out a questionnaire, which was aimed at the technical side of educational institutions. Thanks to which it was possible to analyze the educational system at the present time, and to provide the University of Hradec Kralove with a possible alternative for the introduction of new technology.

Obsah

Seznam obrázků

Seznam tabulek

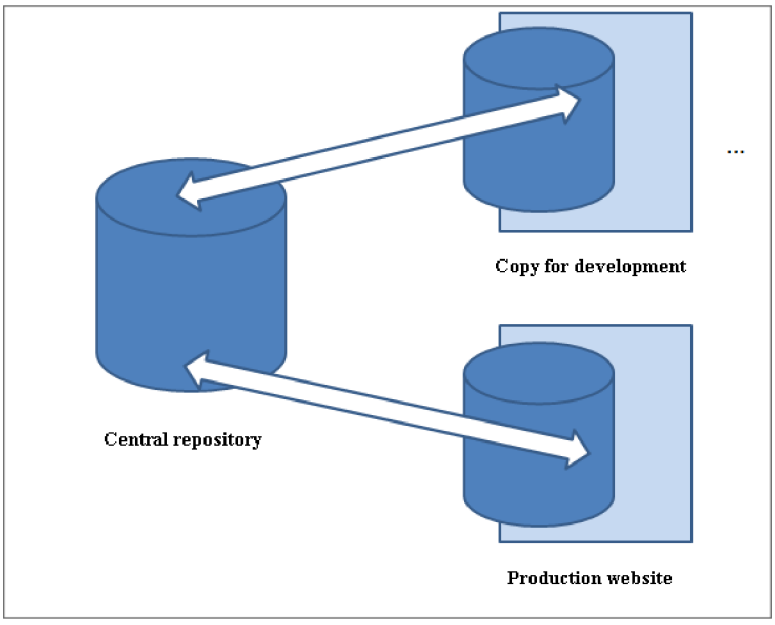
1 Úvod

V současné době, máme kolem nás velké množství komplexních systémů, a přes to musíme pořád opravovat a obnovovat dokumenty v různých fázích vývoje. A po dobu své existence jeden takový dokument může být náchylný k velkému počtu změn. Ale také může se stát, že pro zpracování je nutná nejen poslední verze souborů, ale jedna z předchozích její variant.

V takovém případě můžeme prostě uložit několik různých verzí nutného dokumentů, ale takový způsob je naprosto neúčinní. Protože musíme ztrácet nejen spoustu vlastního času a úsilí, ale také musíme věnovat tomu zvláštní pozornost, s předpokladem že existuje vysoká pravděpodobnost chyb. A v takovém případě dojde k situace, že budeme muset uchovávat obrovské množství téměř identických souborů.

V důsledku čeho začaly zpracovávat speciální softwarové nástroje, který bych mohli zjednodušit celý proces práce s takové dokumenty. V dnešní dobu takové nástroje jsou označovány jako systémy pro správu verzí (Version Control Systém, VCS). Existuje několik systémů tohoto druhu, z nichž každý je relevantní a však má své vlastní specifické rysy v sadě příkazů, způsobů práce a správě. Ale pořád celkový pracovní řád pro většinu VCS je však naprosto stereotypní a předpokládají jednoduchý postupy. Proto z hlediska uživatelů je předpokládáno, že projekt se kterým on chce pracovat, již existuje a jeho repozitář je umístěn na speciálním serveru, kam vývojář obdrží přístup.

Hlavním cílem takových systémů je ukládat zdrojové kódy při vývoji softwaru a jakýkoliv jíních program, a byly zaměřeni jen na IT firmy. VCS umožňuje vývojářům zapojit se ke spolupráce s projektem a také provádět různé manipulace s ním, jako ukládat předchozí verze souborů z vývoje a dostat je odtamtud. Do tak zvaného repozitáře ukládá informace o verzi každého souboru, se všemi změnami a k tomu i kompletní strukturu celého projektu. Ale vzhledem k tomu že tyto systémy zaměření na často se měnící elektronický dokumenty, lze ji použít v jiných oblastech například v CAD (Computer Aided Design), jako součást systémů správy dat nebo také v nástrojích pro správu konfigurace.



Obr. 1 Jednoduchý diagram repozitářů

Zdroj: training.bitrix24.com

V jakémkoliv případě nyní využití VCS je pro vývojáře nezbytné, protože práce nad projektem předpokládá neustálý práce s kódem a jeho úpravu a spolupráce s ostatníma vývojáři. A už je jasné proč využití systému správy verzí je rozhodně lepší než použití úzce zaměřených metod.

V současné době systémy pro správy verzí umožňují řešit řadu standardních problémů:

- ukládání verzí souborů;
- možnost získat všechny předchozí verze uložených souborů;
- zobrazení změn provedených mezi požadovanými verzemi;
- uložení a zobrazení autorů a komentářů k provedeným změnám.

2 Cíl práce

Cílem této práce je seznámit se s principem fungování technologie Git a jejích produktů GitHub a GitHub Classroom, které najednou získaly důvěru běžných uživatelů a vývojářů. Jmenovitě jejich implementaci v oblasti vzdělávání a úvaha nad využitím této technologie na Univerzitě Hradec Králové. K dosažení tohoto cíle je nutné zvážit následující body:

- Podrobně prozkoumat technologii práce se systémy pro správu verzí a jejich schopnosti
- Zvážit postupy pro používání hostingových serverů ve vzdělávacích institucích
- Důkladně prostudovat výukovou platformu GitHub Classroom a zvážit alternativy pro její praktickou aplikaci
- Provést dotazník, mezi studenty vysokých a středních škol o jejich znalostech systému správy verzí, který by zhodnotil současnou situaci moderního systému vzdělávání
- Na základě analýzy navrhnout strategii využití této technologie na Univerzitě Hradec Králové

3 Koncepce systému správy verzí

Systém správy verzí je software, který umožňuje vytvářet neomezené množství verze souborů a pracovat s těmito verzemi jako s nezávislými prvky.

Při práce s VCS můžete nejen vytvořit samotné verze, ale také můžete zvolit strukturu pro jejich skladování. Nyní nejoblíbenější jsou buď řetězy, nebo stromy. Závisí na tom, jak velký máte vývoj a kolik vývojářů momentálně pracuje s projektem. Ale než začít proces práce s elementy a jejich verzemi, musíte poslat všechny potřebné soubory do samotného systému pro správu verzí a společně s elementy se vytvoří i jeho první verze (main). Uvnitř systému správy verzí mohou být jednotlivé prvky umístěny různými způsoby – a to už záleží na zvolené vámi architektuře VCS. Pro vývojáře je důležité vědět pouze, že všechny soubory jsou umístěny uvnitř repozitáře a práce s ním se provádí pomocí příkazů zvolené platformy.

Co se týče struktury pro ukládání verzí, tak nejvíc používanější strukturou je takzvaný „strom“, který může být přímý, v tomto případě se každé následující ukládání souborů provádí po předchozím, v takové situace předpokládá že máte malý vývoj a všechny úpravy dělá jenom vedoucí projektu a rozvětvené. Když říkáme o rozvětvené, tak uvnitř repozitáře ukládá několik paralelních vývojových linií, běžně nazývaných větve. To může být užitečné, když nad projektem pracuje několik vývojářů, v takovém případě můžete uložit stabilní verze která bude základna pro všechny, a dál vývojáři samostatně vytvářejí verze větve jenom pro sebe a mohou zároveň pokračovat v práci.

Obr. 2 Příklad větvení

Zdroj: leanpub.com

Jakmile pracujete nad projektem ve skupině, může se stát, že potkáte řadu problém, a to může být cokoliv. Když několik lidé současně provádějí změny v různých částech dokumentu, tak budete muset zamyslet nad tím, jak se budete provádět sjednocení. „Systémy pro správu verzí pracují s těmito druhy problémů a mají řadu řešení. Ve většině případů tyto systémy mohou automaticky kombinovat tento druh změn, které dělají různí členové vývojového týmu. Je však třeba poznamenat, že tento typ sloučení se nejčastěji provádí u textových souborů a za určitých podmínek: ke změnám došlo v různých částech souborů. K tomuto omezení dochází, protože většina systémů pro správu verzí je zaměřená na podporu procesu vývoje softwarového produktu a původní kódy jsou v textových souborech. V případě, že provedené automatické sloučení selhalo, systém navrhuje opravit situaci ručně. Často se stane, že není možné provést sloučení ani pomocí systému, ani ručně. Zářným příkladem je situace, kdy je formát souboru velmi složitý nebo neznámý. V takovém případě jednotlivé systémy pro správu verzí umožňují uživateli uzamknout soubor v úložišti. Tato operace zabrání ostatním uživatelům získat pracovní kopii, nebo brání změně pracovní kopie souboru a tím poskytuje výjimečný přístup pouze tomu uživateli, který pracuje s dokumentem.“ [41]

Také při práci v týmu může velmi snadno dojít k mírnému nedorozumění ohledně umístění verzí a která z nich je v současné době hlavní. Samozřejmě vždy máme main větev, která je hlavní, ale jak pochopit, na čem každý vývojář pracuje a jak se zapojit do jeho práce, když je to nutné. Pro tyto účely mají systémy pro správu verzí mechanismus značkování / „tagging“. Tag je alfanumerické označení, které jedinečně identifikuje konfiguraci. Tímto způsobem můžete přidat značku do jakékoli verze projektu, čímž usnadníte proces jeho vyhledávání a umožní rychle cílit na to, co se přesně děje v této verzi projektu.

A každý rok se možnosti využití těchto technologií otevírají mnoha profesím a stejným způsobem je možné ukládat práci webových návrhů a grafických prvků. Algoritmus práce zůstává stejný, jediným rozdílem je formát dokumentu, ale na to m moc nezáleží. Stejně jako u kódu můžete ve svých pracích provádět změny, uklád at je a vrátit se ke starším šablonám. Jediným bodem, který může být problémem, j e srovnání, protože formát obrázku a textu se mírně liší.

Systémy pro správu verzí vám tedy poskytují platformu, která si pamatuje a ukládá všechny soubory, které jste jí poskytli, s přihlédnutím k historii vašich akcí, komentářů, problémů atd. A také můžete najít mnoho pluginů třetích stran, díky nimž bude celý zážitek pohodlnější v závislosti na vaší pracovní oblasti.

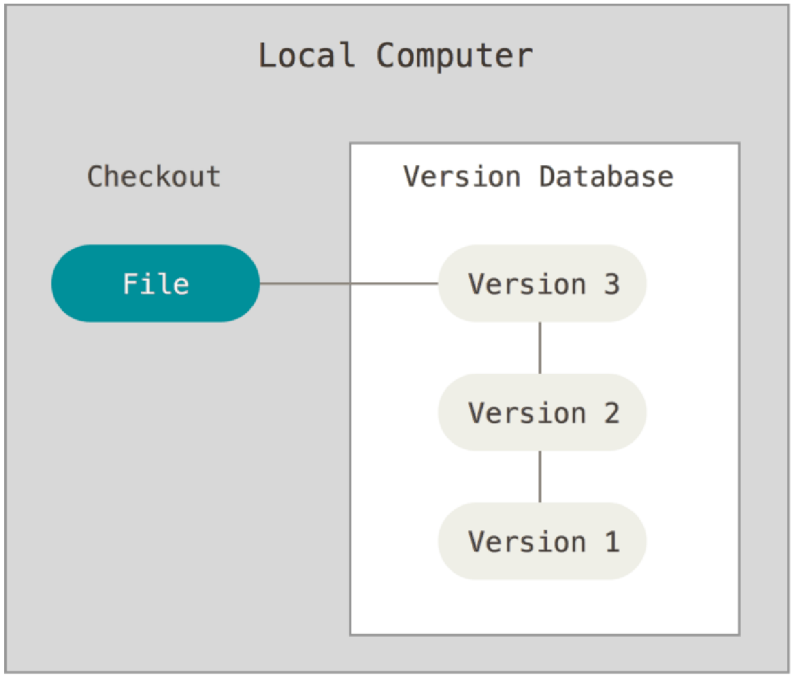
Některé z nejlepších příkladů VCS jsou:

- Git;
- SVN;
- Mercurial;
- CVS (Concurrent Versions System);
- Team Foundation Server.

3.1 Lokální systémy správy verzí

Pokud mluvíme o nějakém malém projektu, který je maximálně určen pro domácí vývoj a nezajišťuje účast třetích stran, pak je nejlepší volbou použít lokální typ systému pro správu verzí. V tomto případě to bude nejbezpečnější možnost, protože databáze je velmi jednoduchá a nevyžaduje zbytečné nastavení.

Když používáte lokální VCS, všechny soubory se ukládají na vašem osobním počítači nebo přenosné jednotce. Podle toho na základě databáze, kterou jste vytvořili. obdržíte seznam ze starší verze projektu do nové.



Obr. 3 Lokální správa verzí

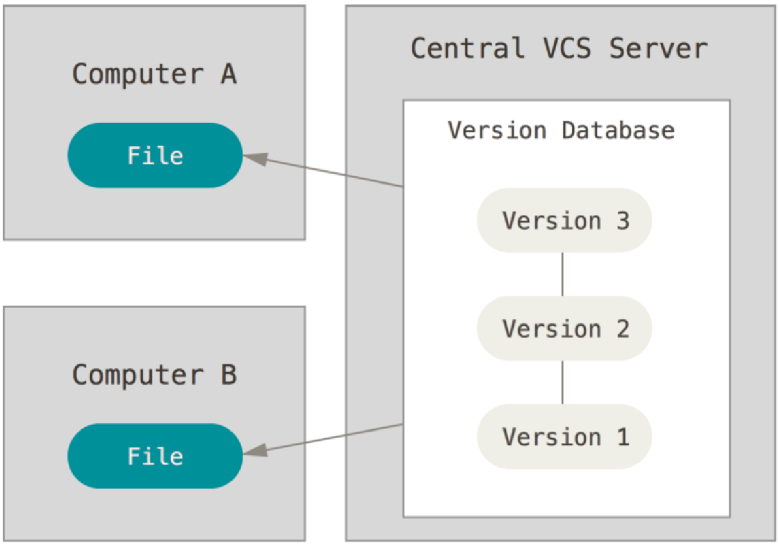
Zdroj: git-scm.com

Tato metoda je docela populární, zvláště když mluvíme o studentech. A je to opravdu pohodlné, protože rychlost práce závisí pouze na vašem počítači, nikoliv na

síťovém připojení. Také není nutné se seznámit s novým rozhraním a nuancemi práce, protože proces je velmi dobře známý. Existuje však několik nevýhod, které mohou zkomplikovat práci, například možnost ztráty dat v důsledku výskytu fyzikálních poruch zařízení nebo nedostatek možností společného vývoje.

3.2 Centralizované systémy správy verzí

Tento typ systému pro správu verzí je již navržen pro vývoj velkých projektů a jak název napovídá, jeho hlavním principem činnosti je ukládání celého projektu na centrální server, který může na vyžádání odesílat soubory.



Obr. 4 Centralizovaná správa verzí

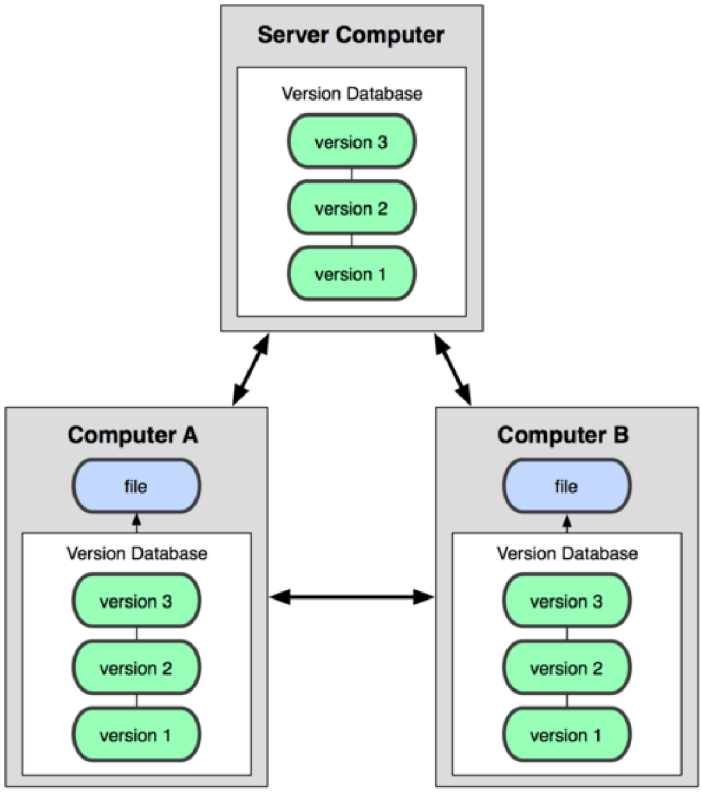
Zdroj: git-scm.com

Celý proces práce spočívá ve skutečnosti, že vývojáři mohou pracovat pouze s nejnovější verzí projektu, protože je přední a tento typ systému pro správu verzí umožňuje lineární vývoj. Samozřejmě existuje přístup k předchozím verzím, ale k tomu budete muset osobně požádat o server, což mírně komplikuje celkový pracovní postup. Nevýhody lze tedy okamžitě připsat poměrně nízké rychlosti práce a přímé závislosti na připojení k internetu, protože bez internetu ztratíte přístup k datům.

Zde je však již jednodušší provádět týmový vývoj (na rozdíl od lokální rozvoje) a používají se nové technologie, které práci zjednodušují.

3.3 Distribuované systémy správy verzí

Distribuované systémy, jako je Git, je moderním modelem procesu celkového vývoje, protože se používají pouze nové technologie. A hlavním mechanismem práce je, že každý, kdo pracuje s konkrétním projektem, má vlastní osobní repozitář, které je uloženo na serveru a na místním počítači. Tímto způsobem nemusíte být neustále připojeni k síti. Právě zde máte schopnost neustále pracovat s větvený a v případě osobní potřeby vytvořit samostatnou verzi projektu, na kterém budete pracovat. Také v tomto případě, když chceme stáhnout data z repozitáře serveru, se stahují všechny uložené změny, nejen nejnovější verze, ale tato operace proběhne pouze jednou, takže pokud chceme aktualizovat data projektu, stačí pouze zkopírovat ty změny, které nejsou v místním uživatelském projektu.



Obr. 5 Distribuovaný systém pro správu verzí

Zdroj: git-scm.com

A opět, díky osobnímu repozitáři, se vývojář může kdykoli vrátit zpět do požadované verze projektu, přepínat mezi větvemi nebo sloučit několik paralelních větví do jedné, která zase bude obsahovat všechny provedené změny. Všechny tyto manipulace se provádějí bez použití hlavního serveru, což výrazně usnadňuje a dělá

práci se skupinovými projekty příjemnější. Na závěr, když jsou všechny větve připraveny k použití a zbaveny chyb, se provádí nahrání změn do hlavního úložiště.

Na druhou stranu se může zdát, že s takovým stylem práce nejsou téměř žádné rozdíly mezi touto metodou a centralizovanou metodou a je ještě obtížnější ovládat všechny procesy. I když ve skutečnosti tak to není, hlavními výhodami distribuovaných systémů je jejich flexibilita a mnohem větší, ve srovnání s centralizovanými systémy, autonomie jednotlivých pracovišť. Počítač každého vývojáře je ve skutečnosti nezávislý a plně funkčním serverem. Z těchto počítačů můžete vytvořit libovolný systém ve struktuře a složitosti, a to zadáním požadovaného pořadí synchronizace.

Pokud jde o nedostatky, jedná se o poměrně vysoké požadavky na objem paměti, protože existuje potřeba uložit celý projekt.

4 Technologie GIT, služba GitHub

Napsali jste kód, spustili ho a funguje to tak, jak by mělo. Přidáte novou funkci a všechno přestane fungovat. Každý vývojář je obeznámen se situací, kdy ho technologie nechce poslouchat. Nic není dokonalé a někdy se něco rozbije. A v takové situaci může hledání malé chyby trvat několik hodin. V takových případech přicházejí na pomoc systémy správy verzí.

4.1 Technologie GIT

Jakkoli to může znít divně, vývojová fáze technologie Git začala jádrem Linuxu. V té době slibný vývojář Linus Torvalds otevřel rozsáhlý projekt, který vyžadoval neuvěřitelné úsilí a použití moderních technologií. A právě proto byla během procesu vývoje použita technologie systému pro správu verzí, konkrétně BitKeeper. V roce 2005 byl však vývoj zastaven, protože vývojář BitKeeperu zrušil bezplatnou verzi svého programu, což způsobilo diskuse o tom, jak pokračovat v práci na na tak rozsáhlém projektu.

V roce 2005 existovalo jen velmi málo alternativ a samotný výběr nebyl příliš velký. A právě z tohoto důvodu je nejlepším řešením bylo vyvinout svůj vlastní systém kontroly osobní verze. A přesně tak, 3. dubna 2005, začal proces vývoje celosvětově populární technologie Git.

Obr. 6 Logo distribuované systému GIT

Zdroj: git-scm.com

„Git je flexibilní, distribuovaný (bez jediného serveru) systém řízení verzí, který poskytuje spoustu příležitostí nejen pro vývojáře softwaru, ale také spisovatelům, kteří mohou měnit, doplňovat a sledovat změny v „rukopisech“ a dějových liniích; učitelé pro přizpůsobení a rozvoj kurzu přednášek; administrátory pro údržbu dokumentace a pro mnoho dalších oblastí, které vyžadují řízení historie změn. Každý vývojář používající Git, má svůj vlastní místní repozitář, umožňující

lokální spravování verze. Pak data uložená v místním úložišti můžete sdílet s ostatními uživateli. Často při práci s Git se vytvářejí centrální úložiště, se kterými se synchronizují další vývojáři. Hlavním příkladem uspořádání systému s centrálním úložištěm je projekt vývoje jádra Linuxu. V tomto případě všichni účastníci projektu vedou svůj místní vývoj a volně stahují aktualizace z centrálního repozitáře. Když je dokončena a odladěna nezbytná práce jednotlivých účastníků projektu, po potvrzení správnosti a aktuálnosti práce provedené vlastníkem centrálního úložiště, se nahrají změny do centrálního repozitáře.“ [44]

Protože je Git distribuovaným systémem řízení verzí, je považován za jeden z nejspolehlivějších systémů svého typu, protože je téměř nemožné ztratit soubory projektu, když projekt je uložen v lokalním počítači každého vývojáře. Jak již bylo zmíněno výše, existuje možnost mnoha paralelních větví, díky nimž je projekt velmi snadný a nezávisí na fázi vývoje.

4.2 Služba GitHub

Jak jsme zjistili výše, Git je nástroj, který umožňuje implementovat distribuovaný systém kontroly verzí. GitHub je vývojová platforma, kterou vývojáři používají k ukládání projektů — odtud název hub[1].

Myšlenka na vytvoření GitHubu byla sponzorována vývojovým týmem - Tom Preston-Werner, Chris Vantrath a PJ Hyatt - a v roce 2008 vytvořili neuvěřitelný nástroj, který dodnes zůstávají na vrcholu popularity. A jeho použití otevírá lidem další a další nové obzory, protože máte nejen bezpečné místo pro ukládání svých zdrojů, ale také schopnost pracovat s ostatními lidmi po celém světě bez omezení. Tak se tyto technologie každý rok stávají nedílnou součástí velkých developerových firem i malých podnikatelů.

[1] Hub – the central part of something, esp. of a wheel, or a center of aktivity.
Meaning of hub in English. In: Cambridge Dictionary[online]. c2020, 24. 08. 2020 [cit. 2021-04-29]. Dostupné z: ttps://dictionary.cambridge.org/dictionary/english/hub

GitHub lze považovat za druh sociální sítě, která umožňuje nejen udržovat komunikaci mezi vývojáři, ale také obsahuje všechny potřebné nástroje pro správu kódů. „Běžní“ uživatelé mohou sledovat ostatní vývojáře a dostávat aktualizace aktivit, získávat informace o nových projektech a trendech, učit se od vývojářů a také provádět spolupráci. Z GitHub se vývojáři mohou snadno odhlásit a odstranit všechny uživatele ze svých seznamů, což může také omezit přístup k jejich vývoji a projektům.

Obr. 7 Logo GitHub

Zdroj: siliconangle.com

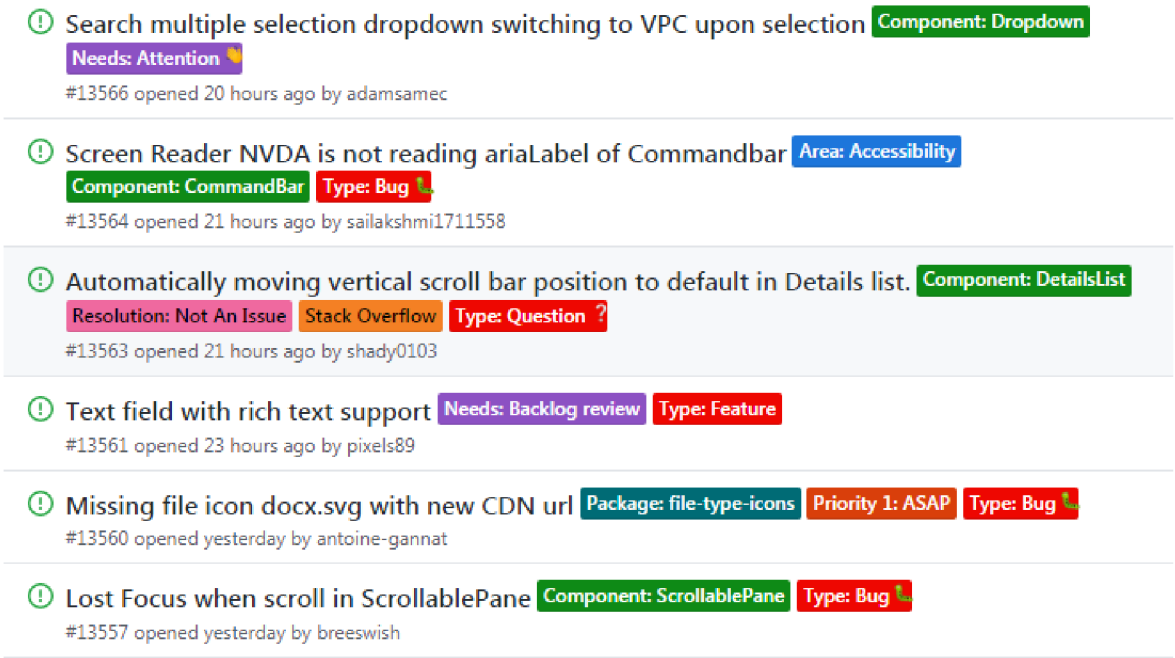
GitHub obsahuje neuvěřitelné množství funkcí a aktivit, které pomáhají vývojářům po celém světě se podílet na projektu. Vývojáři mohou psát zprávy o problémech, aby identifikovali chyby, dokumentovali softwarové kódy nebo vylepšili software psaním požadavků na funkce. Vývojáři vytvářejí kopii repozitáře a provádějí změny k implementaci nových funkcí nebo oprav chyb; zasílají "pullrequests" pokud chtějí sloučit změny kódu do hlavního úložiště. A pouze hlavní účastníci projektu mohou schvalovat a povolovat změny kódu v hlavním repozitáři.

4.2.1 GitHub Issues

GitHub Issues je jedním z nejpopulárnějších systémů pro sledování chyb na světě, který poskytuje vlastníkům repozitáře schopnost organizovat, označovat a propojovat problémy s konkrétními fázemi vývoje.

Původně byla tato funkce zaměřena na open-source, protože spolupráce, která je podporována po celém světě, zní jako neuvěřitelná vlastnost. Ale zároveň dokonale zapadá do komerčních projektů. Tuto funkci lze použít jakýmkoli způsobem, můžete pouze otevřít běžnou diskusi, kde se mohou ostatní uživatelé připojit k konverzací nebo vytvořit úkol pro vývojáře projektů.

V sekci "issues" každý projekt obsahuje úplné informace a stručný popis zjištěných problémů. Vzhledem k tomu, že otázky může klást kdokoli, v důsledku toho, nemusí být úroveň detailů vždy dostatečná k pochopení problému nebo požadavku. Stává se tak, že odpovědi na vaše otázky přicházejí okamžitě. A někdy problém zůstává nevyřešen měsíce. Během této doby obdržíte připomínky jiných vývojářů a v budoucnu se k němu můžete vrátit, abyste něco opravili nebo vylepšili kód.



Obr. 8 Příklad GitHub Issues

Zdroj: github.com

4.3 Praktická aplikace GitHubu ve výuce

V současných reáliích se GitHub a podobné funkční servery stávají něčím více než jen místem pro ukládání kódu, například když se vezme v úvahu situace s pandemií, takové virtuální kanceláře se stávají místem setkání, které podporuje spolupráci. A podporuje vzájemnou pomoc, diskusi a sdílení informací mezi jeho komunitou.

To je důvod, proč je tato platforma najednou považována za druh fóra (jako Stack Overflow), které funguje na základě vzájemné pomoci. A tento názor má hodně co říct, protože v nejdřívější verzi existovali lidé, kteří začali používat GitHub mírně nekonvenčním směrem, což vytvořilo tento názor. Koneckonců, když mluvíme o systémech VCS, každý průměrný člověk si představí proces vývoje softwaru. Jiní

však viděli perspektivu užívání jako sociální síť, a takto se na otevřených prostorech serverů Git objevila úložiště, která byla určena pro úpravy písma, skládání hudby a ukládání právních dokumentů. A tak se tato technologie postupem času stala něčím víc než jen webem pro vývoj. Nakonec si lidé začali vyměňovat recepty, aby dosáhli ideální chuti a vyměnili si společný zkušenosti; jiní publikovali zákony a předpisy federálního zákoníku z různých zemí, které umožňovaly komukoli sledovat změny a různá zdůvodnění změn.

Ale co se týče aplikace pro vzdělávací účely? Samozřejmě, jak jsme již zvážili výše, celkové využití této technologie pro účely učení existuje a rychle se vyvíjí. A každým rokem (jako nová generace učitelů nahrazuje starou) roste motivace a touha změnit tradiční systém řízení učení. Pokud se pokusíme zvážit hlavní motiv a účel tradičního systému, jedná se o výměnu znalostí a vzdělávacích materiálů učitele se studenty a poskytování praktických úkolů studentům (s uvážením na jeho ověření). Skutečně, kdysi neexistovaly žádné moderní technologie, bylo důležité, aby studenti navštěvovali vzdělávací instituci, aby si mohli dělat osobní poznámky a rozšířit své znalosti, protože najít ten správný materiál byla velmi obtížná a časově náročná práce. V moderní době však máme kombinaci mezi tradičním systémem vzdělávání a zaváděním moderních technologií pro pohodlnou výměnu informací (i když dodnes je upřednostňován pouze e-mail). Kromě toho přístup k materiálům nebo úkolům kurzu je k dispozici pouze současným studentům. To nezahrnuje studenty, kteří absolvovali kurz a chtějí materiál používat opakovaně. A nezahrnuje ani další učitele nebo zástupce, kteří se mohou podělit o své vlastní zkušenosti.

4.3.1 Benefity

Samozřejmě, že přechod na platformu GitHub sám o sobě nevyřeší všechny problémy, ale tato alternativa má své výhody. A když používání technologií Git ve výuce přestalo být něčím super novým, lidé začali pracovat na tom, co by učinilo obsah přístupnější pro všechny, v důsledku čehož se objevil takový koncept jako otevřený vzdělávací zdroj (Open Educational Resources, OER). To znamená, společné zdroje pro výuku, vzdělávání a výzkum, dostupné na právně otevřených licencích, které mohou lidé zdarma znovu použít, upravit, remixovat a distribuovat.[35] A mnoho mladých učitelů takovou iniciativu podpořilo, protože

téměř okamžitě po příchodu tohoto termínu se na rozsáhlých fórech hodně diskutovalo a zároveň se na serverech objevily první publikace vzdělávacích materiálů (včetně akademického plánu).

Jedním z příkladů je projekt „Coding Train“[2], který nemá osnovy, ale jedná se o kompletní kurz nahraný na GitHub a obsahuje články, vzdělávací videa a související materiály, které mohou být vynikajícím průvodcem při učení Machine Learning. Nebo plnohodnotný projekt vysokoškolského učitele - „The Nature of Code“[3]. V tomto příkladu již vidíme projekt s kompletním učebním plánem na celý semestr, kde se každý může stát součástí vzdělávacího procesu. Tento kurz obsahuje nejen podpůrné materiály, ale i praktické úkoly. Oba projekty mají více než 200 rozvětvení, probíhají otevřené otázky (issues) která zahrnují nejen iniciativu předních projektů, ale také návrhy zvenčí (studenti a běžní uživatelé).

Učební plán lze považovat za nejdůležitější součást při navrhování vzdělávacího systému. A stejně jako v každém procesu, kde je sázka na zlepšování, je velmi důležité sledovat změny a neustále je porovnávat, aby se zabránilo opakovaným chybám. A právě takové platformy jako GitHub poskytují transparentnost verzí a během každého semestru umožní zlepšit výběr materiálu kurzu a bude odrážet jejich vývoj.

Samozřejmě, že systémy pro správu verzí nebyly původně vytvořeny pro vzdělávací účely, ale od roku 2010 pedagogové a lidé, kteří viděli potenciál v Git-hostování, začali aktivně testovat takovou implementaci. Takové experimenty samozřejmě nebyly okamžitě korunovány úspěchem, protože v té době bylo nastavení místního serveru pro běžné učitele velmi náročným úkolem a vyžadovalo poměrně vysokou úroveň znalostí v oblasti IT. Teprve v roce 2013 experimenty s GitHub přinesl první pozitivní recenze, a to nejen díky jeho funkčnosti, ale především díky bezplatným úložištím. A o rok později, když vedení GitHub zaznamenalo velké množství studentských účtů (které přesáhlo 50 000), GitHub veřejně oznámil, že spouští speciální vývoj zaměřený samostatně pro vzdělávací

[2] https://github.com/CodingTrain/Machine-Learning#articles--posts
[3] https://github.com/nature-of-code/NOC-S17-2-Intelligence-Learning

účely, a zavedl bezplatné výhody pro studenty a vzdělávací instituce. Tuto iniciativu převzaly další hostingové platformy.

Takovým způsobem, mnoho Git-hostingových platforem, jako jsou BitBucket, GitHub a GitLab se začaly aktivně angažovat v propagaci svých platforem. Jmenovitě šíření výše zmíněných bezplatných repozitářů mezi učiteli a studenty, kteří už nemohli potlačit svůj osobní zájem. A právě proto lze tento výchozí bod považovat za rozhodující nejen pro tuto technologii, ale i pro radikální změny ve vzdělávací oblasti. Ale také bylo často vytýkáno, že technologie git-hostingu je velmi spolehlivá pro řízení procesu učení a navzdory mírně složitým nástrojům pro spolupráci, přítomnost přímé komunikace studenta a učitele urovnal problémy s řízením, protože vždy existuje příležitost k přímé konzultaci (které se mohou zúčastnit i ostatní studenti). Samozřejmě, nikdo nemůže zajistit hladké zavedení této technologie a nelze vyloučit, že obě strany budou čelit obtížím s jejím používáním, nicméně skutečnost, že v dnešní době je schopnost pracovat s VSC a hostingem je prostě nutná.

Ale pokud jde o uvedení jakýchkoli informací do veřejné domény, je velmi důležité zachovat důvěrnost a chránit integritu souborů. A i když populární servery již poskytly bezplatné repozitáře pro vzdělávací instituce, zdá se, že už není možné udělat nic lépe, také vytvořily ideální podmínky pro ochranu a důvěrnost. A to samozřejmě ovlivnilo vytvoření osobního názoru na tuto technologii ze strany učitelů i administrátorů, protože si mohli být jisti, že studenti budou moci získat plnou ochranu. Samotní developeři to považovali za příspěvek do budoucnosti, protože čím více zainteresovaných stran, tím větší bude příspěvek do společnosti a zaměstnancům v blízké budoucnosti.

Ale i kdyby ano, je těžké uvěřit, že pouze bezplatné repozitáře a úplná důvěrnost přinesly Git-hostingu takový úspěch. A vyvstává otázka, co se stalo, že se po třech letech zcela nevhodný nástroj pro učení najednou se stal "hitem"? Hlavním klíčem byli učitelé technických univerzit, kteří se stali dobrovolníky pro vývoj individuálních nástrojů. Čím více pozornosti to přitahovalo, tím více lidí mělo zájem, aby vytvořili svůj vlastni příspěvek. Postupem času se objevila spousta dalších programů, které umožňovaly řešit velké množství problémů s administrativními úkoly, a samozřejmě vše bylo open source. Právě díky tak aktivní komunitě GitHub,

která neměla žádnou funkčnost pro řízení vzdělávání, konečně dostala příležitost pojmout všechny potřebné nástroje, a to - „distribuce pracovních úkolů, shromažďování úkolů k hodnocení, přiřazování úkolů vzájemného hodnocení, propojení s nástroji třetích stran pro automatizaci úkolů hodnocení a sdílení zpětné vazby se studenty."[21] Nejpozoruhodnějším příkladem je RepoBee[4]. Samotný produkt byl vytvořen učiteli švédské univerzity, aby se technologie Git, co nejvíce přiblížila pro práci ve školách. Samotný nástroj podporuje a splňuje standardní potřeby učitelů – distribuci a přijímání úkolů, vzájemné hodnocení, zpětnou vazbu, nástroje třetích stran, jednoduché rozhraní příkazového řádku pro učitele. A i když taková vyhlídka zní mnohem přitažlivěji, tomuto produktu chybí grafické rozhraní a jakýkoli druh systému hodnocení. Pro učitele technických směrů příkazového řádku však bylo více než dost, protože se základními znalostmi bylo možné vykonávat mnoho funkcí. Koneckonců se jedná o nástroj, který obsahuje 20 příkazů, z nichž některé jsou určeny pro:

- Konfigurace
- Správa jednotlivých sfér (repozitář, issues, plug-in atd.). Například pokud chceme aktualizovat úložiště repos update. Ale také stojí za zmínku přítomnost několika chytrých příkazů, například jako open-issues a close-issues(otevření a zavření problému), což vám umožní pracovat rychle a pohodlně.
- Provádění akcí (clone, update, setup). Takové příkazy jsou již považovány za standardní a podle názvu je velmi snadné zachytit jejich záměr – klonování repozitářů studentů/učitelů; aktualizace repozitáře; a přímo vytvářet repozitáře/pluginy/problémy.

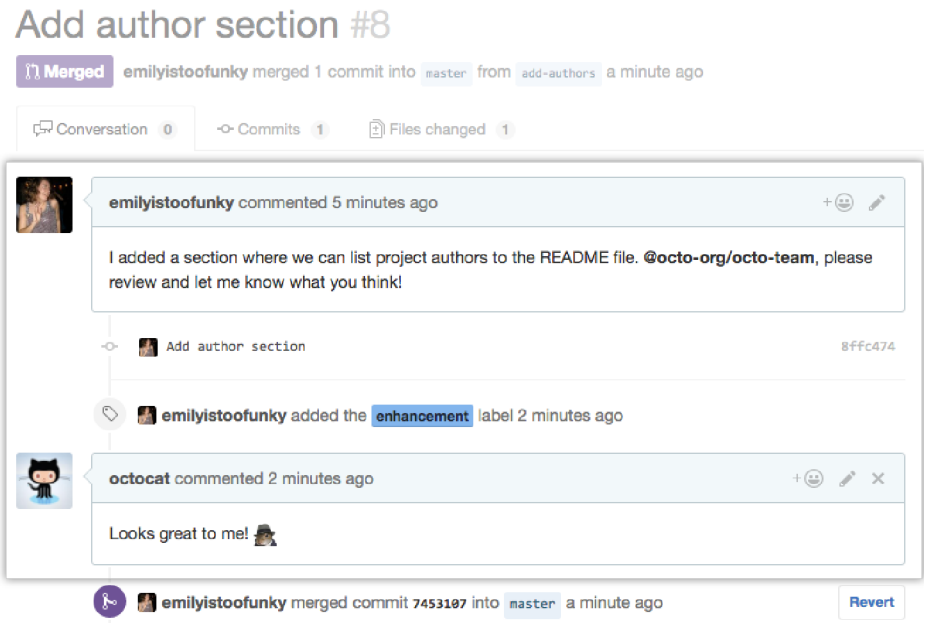
Tím ale nástroje tohoto druhu nekončí, protože podobnou iniciativu převzali i další nadšenci. A proto můžete vytvořit celý samostatný seznam takových nástrojů, které mají mírně odlišnou funkčnost, ale podstata se nemění. Například nástroj Gitomator[5], který byl vytvořen pro provádění automatických hodnocení, ale bohužel

[4] https://repobee.org/
[5] https://gitomator.github.io/

zůstal v pre-alfa fázi. Některé z projektů se do hotové verze vůbec nedostaly a byly uzavřeny. A i když se podíváte na produkt GitHub Classroom, lze jej samozřejmě označit za favorit, ale existují i lidé, kteří se domnívají, že sbírka nástroje pro běžnou verzi GitHub je mnohem pohodlnější.

Co se týče zpětné vazby, otázka je v určitém smyslu trochu diskutabilní. Na jedné straně máme skutečně příležitost rozšířené spolupráce a komunikace jak mezi studenty, tak s učitelem. Když podrobněji zvážíme možnost tohoto sdělení, tak nekončí pouze elementárními komentáři. V každé práci může učitel i student zanechat README dokument, který v sobě bude obsahovat všechny podpůrné informace, což je také zdrojová dokumentace. Nebo jako varianta Wiki, která obsahuje všechny centralizované informace a dokonce i učební osnovy. A samozřejmě, standardní recenze a sledovače problémů (Issue Trackers), ve kterých jsou zaznamenány všechny problémy, které lze sledovat. Ale když mluvíme o 30 studentech, taková strategie zpětné vazby může způsobit určité komplikace a zmatek.

Brzy GitHub nabídl alternativu pro pokročilejší komunikaci, a to použití požadavků na vyžádání, které by umožnily seskupit tok otázek na učitele. Zpočátku byla tato technologie vyvinuta za účelem uspořádání speciálního diskusního kanálu, který bude zaměřen na specifika dané problematiky v procesu vývoje. V tomto případě však nabídli sdílení návrhů s ostatními studenty a učitelem, což dává možnost podílet se na společné práci na problémech, stejně jako v případě, že někdo jiní v budoucnu dostane podobný problém – řešení bude veřejně přístupné.[15] Tato metoda má také možnost komentovat jednotlivé řádky kódu, respektive když je komentář nebo otázka zavedena ze strany studenta, otevře se druh diskuse, ke které se učitel může připojit, nebo naopak se otevírá otázka ze strany učitele pro studenta.



Obr. 9 Příklad GitHub komentování

Zdroj: docs.github.com

A je zřejmé, že taková neustálá zpětná vazba zvyšuje kvalitu práce a ve vědeckých pracích, které se podílejí na psaní tohoto dokumentu, se více než jednou zmiňují pozitivní recenze o kvalitě této funkce a jejím provádění. Koneckonců, jedná se o jednu věc, když mluvíme o 10 minutách konzultace v průběhu praktické lekci, která nebude vždy stačit na kvalitní práci, je zcela odlišné, když je neomezený čas, který může student a učitel věnovat mimo učebnu.

Posledním nástrojem, který se stal součástí vzdělávacího systému, je kontinuální integrace (Continuous Integration, CI). Hlavním účelem tohoto softwaru je fixovat kód při "push" vyžádání, který spouští automatické testy kódu a vhodně se snaží odhalit možné chyby - „vyřešit konflikty sloučení; zkontrolovat formátování stylu, kontroly zabezpečení, pokrytí kódu, funkční testy a další vlastní kontroly“.[1] Takový systém je velmi užitečný, protože umožňuje detekovat část kódu, ve kterém lze nalézt danou chybu nebo poruchu. Proto, pokud se pokusíme prezentovat situaci, kdy učitel musí zkontrolovat 30 projektů, může takové testování předem poskytnout učiteli i studentovi informace o nějakém druhu poruchy, což oběma stranám ušetří čas. Samozřejmě zde existují určité nuance, protože tento test bude spuštěn až po odeslání projektu do hlavního úložiště, a vhodně, pokud hovoříme o situaci, kdy se student odevzdává práci v termínu, pak není moc času vyřešit

problém. Ale samozřejmě na takový problém existuje již řada serverů třetích stran, který umožňuje provádět testy samostatně (CircleCI, Appveyor, Jenkins atd.).

Pokud jde o detekci plagiátorství, v Git-hostingu tato fáze stále pracuje na intuitivní úrovni. Samozřejmě nemůžeme zrušit funkci GitHub, která vám umožní sledovat všechny změny v úložišti, protože je jasné, že za jeden den vytvořit krásný funkční kód pro studenta, který není příliš aktivní po celý semestr, je nemožný úkol. Proto směs zpětné vazby, sledování postupu práce a provedených změn dává učiteli obraz každého studenta o jeho aktivitě. Neexistuje však žádný holistický nástroj pro analýzu textů pro plagiátorství.

5 GitHub Classroom

„Softwarové aplikace jsou obrovským množstvím řádků kódu, které jsou vytvářeny pomocí editorů kódu, integrovaných vývojových prostředí (IDE). Úspěšný vývoj aplikací je doprovázen tvorbou dokumentace s výhradou neustálých a nekontrolovatelných změn, které jsou provedeny mnoha účastníky projektu. Během vývoje projekt prochází změnami jak ve směru kódování nových dílů a modulů, tak při vytváření nových verzí stávajících částí. Práce na projektu by měla být monitorována a samotný projekt upraven tak, aby vývojový tým koordinoval své činnosti a měl jasnou představu o tom, co se děje s verzemi aplikace, a které části byly změněny. Jak již bylo zmíněno vícekrát, systémy správy verzí (VCS) jsou základními nástroji pro vývojáře softwaru. Aby bylo možné porozumět realitě moderního IT průmyslu, je třeba studenty počítačových specializací seznámit se s VCS, a také vytvářet u studentů potřebné dovednosti pro práci se systémy kontroly verzí, jako jsou Git a GitHub. Systémy správy zdrojového kódu mají v softwarovém inženýrství velkou hodnotu a každoročně rostou i požadavky na znalosti o moderních technologiích a jejich využití (a to se netýká pouze VCS). Postupem času rychle rostoucí společnost GitHub převzala dominantní postavení ve světě vývoje aplikací, který má plně nakonfigurované nástroje pro správu verzí. První věc, která přitahovala zájem uživatelů, bylo to, že web GitHub umožňoval programátorům snadnou interakci přes internet, poskytoval komunikační a sociální funkce, například sledoval určité části kódu v projektech.“[24] Stojí také za zmínku, že mluvíme o začátku roku 2009, kdy si lidé ani nedokázali představit, jaké by to bylo používat k těmto účelům internet.

Vždy stojí za to si uvědomit, že samotný proces učení není nejjednodušší praxí a studium informatiky a vývoje softwaru je obtížné z mnoha důvodů. Zatímco studenti se potřebují naučit teorii, musí také aplikovat to, co se naučili, na praktickém nastavení. Nejlépe se učí v praxi tím, že dělají chyby a opakují rozhodnutí. A protože moderní komplexní systémy jsou psány pracujícími vývojáři, podrobné informace o procesu a jeho obecné technologii jsou pro běžné uživatele internetu izolované a stávají se výrobním tajemstvím, proto musí studenti věnovat značné množství času tomu, aby se naučili spolupracovat a koordinovat jejich

jednání s ostatními. Nebo musí věnovat více času na samostudiu, aby byli alespoň o krok napřed, nebo drželi krok se svými kolegy.

A jak ukazuje praxe, GitHub se stále více používá jako výuková platforma díky svým otevřeným funkcím pracovního toku a transparentnosti a také proto, že učitelům nabízí možnost opakovaně používat kombinaci materiálů kurzu (prostřednictvím distribuované kontroly verzí) a nabízí svým studentům příležitost přispět svým příspěvkem do kurzových materiálů (například prostřednictvím požadavků na stažení).

Neexistoval však žádný holistický nástroj, který by byl směrován do školních osnov s jeho vlastními nuancemi, až dokud se student počítačové vědy Mark Tareshawty v roce 2014 nerozhodl zahájit vývoj univerzálního rozhraní pro studenty a učitele. A právě tato myšlenka ho nakonec přivedla do společnosti GitHub, kde mladý a nadějný Mark získal pozici ekosystémového inženýra. Původně jeho hlavním cílem bylo vytvořit takový nástroj, který by mohl výrazně zjednodušit proces "řízení" studentů a úkolů, ve kterém učitelé přihlašují studenty na GitHub – k vytváření a sdílení úkolů v programování. To znamená, že učitel může vytvořit mnoho repozitářů, které budou obsahovat šablony se zdrojovým kódem a pokyny pro práci/projekt. Poté mohou studenti, kteří jsou zahrnuti do seznamu přístupů, stisknout tlačítko, které automaticky vytvoří vlastní kopii repozitářů, čímž vytvoří vlastní přizpůsobitelnou adresu URL, kterou může učitel snadno sledovat a spravovat individuální repozitář každého studenta. A právě s takovými ambicemi začala nová vývojová větev GitHub Education, známá také jako GitHub Classroom.

A téměř okamžitě, jakmile vývoj zašel o něco dále než jen nápad a získal první funkcionalitu, GitHub Education se pokusil, co nejvíce rozšířit své pokrytí a nabídl školám nejen nejziskovější nabídky, ale také se tím pokusil vytvořit kolem nich velký přidružený program, což jim zajistilo neuvěřitelný úspěch. Zorganizovali tedy GitHub Campus Program a aby se mohli stát součástí této komunity, GitHub vyžadovalo:

1. informovat všechna technická oddělení vzdělávací instituce o spolupráci s GitHub;
2. umístit jejich logo na oficiálních stránkách školy;
3. souhlasit s přijímáním pravidelných oznámení od GitHub Education;

4. jmenovat správce, který bude zodpovědný za dokončení vzdělávacího programu pro učitele s kurzem od GitHub.

Na oplátku školy získávaly rozsáhlou podporu od Git. A to není jen školení učitelů k zvládnutí Gít a GitHub, ale také mnoho dalších výhod, jako jsou:

- Bezplatný přístup k serveru GitHub Enterprise Server a GitHub Enterprise Cloud.
- Automatický přístup k funkcím GitHub Education (Student Developer Pack).
- Rozvoj vedení a technické školení studentů v rámci programu Campus Experts.

Právě díky této spolupráci má GitHub také příležitost aktivně shromažďovat data a analyzovat váhu procesů pro zavádění nových technologií do vzdělávacího systému. A dnes na jejich oficiálních webových stránkách najdete úplné zprávy s grafy a podrobnou analýzou[6]. Také nemůžeme nezmínit jejich aktivitu v mnoha aktualizacích a jejich samotné zprávy ve formátu blogu, kde najdete mnoho zajímavých článků o jejich cestě. A to je přesně to, co přitahuje mnoho vzdělávacích institucí natolik, aby se stal součástí této komunity.

Tato technologie je stále v aktivním vývoji a společnost GitHub se nezastaví. V současné době se Issue Tracker tohoto nástroje rozrůstá o 900 uzavřených a 200 otevřených případů, které uvádí aktuální vývoj a podporu platformy; a také téměř 1 400 uzavřených a 22 otevřených požadavků na vyžádání od uživatelů z celého světa[7].

Bohužel během dlouhého vývoje bylo mnoho funkcí a nástrojů v omezeném přístupu (například počet bezplatných repozitářů atd.), a stejně mnoho z nich bylo v procesu opravy. Již v roce 2019 se však příležitosti pro studenty i učitele staly otevřenějšími, což jim umožnilo dosáhnout nových výšin v odborné přípravě.

[6] https://education.github.com/classroom-report/2020
[7] https://github.com/education/classroom

5.1 Analýza prostředí GitHub

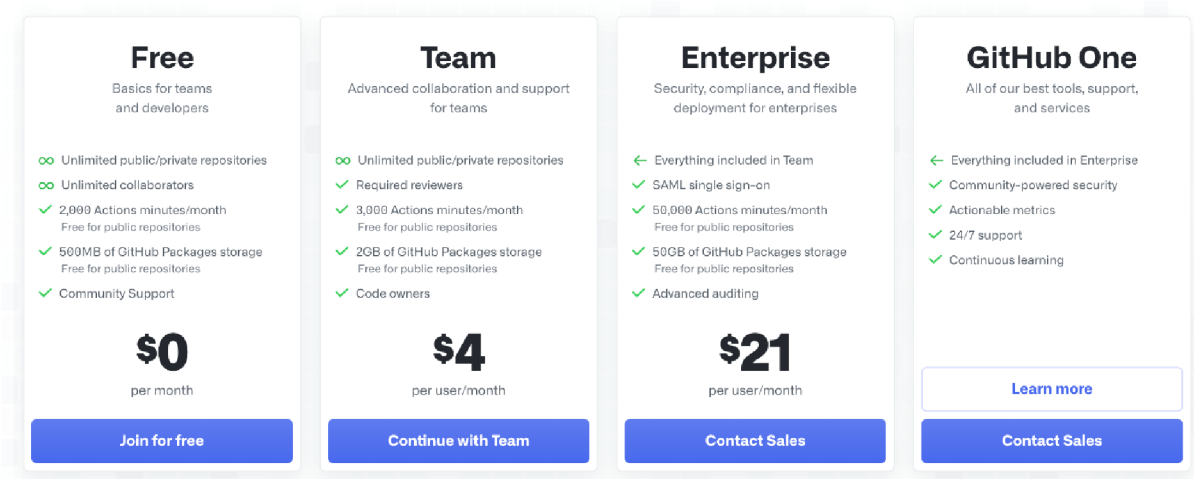
Samozřejmě, když mluvíme o takovém hostitelském serveru, jako je GitHub, vnímáme ho jako samostatnou jednotku, která může fungovat nezávisle. Je holistickým nezávislým projektem s vlastními pravidly a nuancemi. Když narazíme na GitHub Classroom, okamžitě ho nevnímáme jako samostatný nástroj, ale jako součást obecného mechanismu, který pravděpodobně nebude užitečný pro každého, ale bude velmi žádaný pro samostatnou komunitu. Proto, když hovoříme o tom, že začneme pracovat s nástrojem, který je součástí obecného procesu, pak musíme být obeznámeni a rozumět tomu, jak fungují jeho základy. A tak bych nejprve zvážíme všechny možné základní body, které bychom opravdu měli znát, než začneme pracovat s GitHub Classroom.

Je zřejmé, že je GitHub trochu komplexní pro potřeby vzdělání, protože je již delší dobu považován za platformu pouze pro podnikatelský rozvoj. V této chvíli, když už máme nějakou představu o tom, co je GitHub a k čemu slouží, bych ho chtěla zhodnotit z hlediska funkčnosti. Nejprve poznamenat, že zahájení takové cesty jako „samouka“ není nejjednodušší, protože i přes neuvěřitelný technologický pokrok takové platformy zůstávají černými koňmi a vyžadují spoustu času, než se začnete cítit pohodlně při práci s nimi. Proto se mnozí vyhýbají tomu, aby se začali učit, jak s touto technologií pracovat. Ale bez ohledu na to, jak děsivě to zní, nejdůležitější je začátek. Protože hned po prvních krocích se objeví zvědavost, která v tomto případě funguje jako neuvěřitelná motivace. Poté, když jste vedeni touto zvědavostí, moderní technologie a neomezené zdroje internetu vám jdou na ruku. Protože když je výměna informací neuvěřitelně snadná a rychlá, existuje mnoho pomocných listů pro používání GitHub, spousta otevřených konverzací, a dokonce i malí pomocníci, jako je seznam všech příkazů a jejich účelu[8,9].

[8] https://gist.github.com/cferdinandi/ef665330286fd5d7127d
[9] https://www.kutac.cz/serialy/jak-na-git

5.1.1 GitHub tarify

Pokud jde o samotná specifika Githubu, pak zde najdeme několik bodů, které stojí za zvážení. Úplně první věc, na kterou se podíváme, je schopnost vytvořit vylepšený profil. Vylepšením je získání výhod za příplatek.



Obr. 10 Oficiální sazby, které poskytuje GitHub

Zdroj: github.com

Samozřejmě, když mluvíme o vzdělávacím programu, pak půjde řeč vždy o bezplatné verzi, protože studenti a učitelé využívající GitHub Classroom mají svá vlastní privilegia, o kterých budeme podrobně hovořit a analyzovat o něco později. Ale zbytek skupin je již určen pro malé a velké podniky. Vzhledem k tomu, že podmínky, které jsou prezentovány v bezplatné verzi, nestačí pro vývojové prostředí.

První možná privilegia jsou pokročilé funkce pro používání Github Actions. GitHub Actions je funkce, která byla na GitHub zavedena poměrně nedávno a umožňuje spouštět automatizované procesy. Jinými slovy, GitHub vyvinul nástroj, který se stará o řadu okamžiků místo nás a může spustit různé procesy, které mohou být spuštěny standardními událostmi, jako je požadavek na vyžádání, vytvoření nové verze (release) nebo problémů. GitHub tedy poskytuje příležitost nejen k vložení dobrého bezplatného CI/CD (kombinace kontinuální integrace (CI) a kontinuálního nasazení (CD)), ale také k vytvoření velmi flexibilního a snadno konfigurovatelného systému podpory vývoje. Když mluvíme o tak pohodlných nástrojích, které chceme neustále používat při vývoji, zní bezplatný tarif s 2 000 volnými minutami za měsíc velmi slabě.

A samozřejmě cloudové úložiště. V naší době neexistuje žádný server, který poskytuje neomezené úložiště, a GitHub není výjimkou. Tak je to i pro běžný bezplatný profil, a to 500 MB, ale jakmile překročíte tuto hranici, existuje možnost navýšit kapacitu vašeho osobního úložiště.

A také pokročilé funkce, když mluvíme o bezplatné verzi pak se jedná o standardní sadu nástrojů, se kterými se musíme naučit pracovat sami. Ale v tuto chvíli existuje příležitost získat velmi zajímavý tarif, jako je GitHub Pro, který je považován za rozšířenou verzi pro vývojáře a studenty. Má mírně rozšířené úložiště a poskytuje několik doplňků ve formě Wiki, schopnost provádět vícenásobný požadavek na vyžádání a různé grafiky pro analytiku. V tarifu GitHub Team jsou možnosti již velmi příjemné, protože vývojáři mají možnost nastavit si jakákoli připomenutí, vlastní návrhy a mají zvýšenou ochranu. A posledním možným tarifem je GitHub Enterprise. Ten poskytuje neustálou podporu komunity GitHub, další bezpečnostní opatření a otevřený přístup k plnému využití všech výhod GitHub. Existuje také další tarif, konkrétně GitHub One, ale v současné době o něm není mnoho informací. Na oficiálních stránkách je uvedeno, že obsahuje všechny funkce tarifu GitHub Enterprise, ale s vylepšenými metrikami a analytickými zprávami, a také poskytuje příležitost ke zlepšení jejich kvalifikace ve GitHub Learning Lab.

5.1.2 SSH Klíče

Další funkcí je konfigurace klíče SSH. Je pravda, že se nejedná o výjimečnou technologii pro GitHub, ale je navržena tak, aby byl webhosting obecně bezpečnější. Jeho hlavní výhodou je, že správně nakonfigurovaný SSH-klíč vám umožní svobodně pracovat z vašeho počítače. Jmenovitě mít neustálý přístup k projektům uloženým ve službě a provádět příkazy v konzole bez neustálého potvrzování uživatelského jména a hesla.

Každý klíč SSH obsahuje dvojici: veřejný a soukromý klíč. Veřejný klíč se odesílá na server, takže ho nemusíte před všemi skrývat a nemusíte se bát, že ho někdo uvidí a ukradne. Bez dvojice soukromých klíčů je to k ničemu. Ale soukromý klíč je ta tajná část. Přístup k němu byste měli mít pouze vy.

Pokud jde o samotný pracovní proces, probíhá tímto způsobem. Pošlete nějaké informace na server, kde je uložen váš veřejný klíč, server pochopí, že jste to

vy, a identifikuje vás díky soukromému klíči a dá vám nějakou odpověď. A pouze vy můžete dešifrovat tuto odpověď, protože pouze vy máte vhodný soukromý klíč. Ukázalo se, že je to něco jako balíček přihlašovacího jména a hesla, jen mnohem bezpečnější. Vaše heslo někdo může zjistit nebo uhodnout, ale aby získal váš soukromý klíč SSH, útočník se bude muset nabourat do vašeho počítače.

Chcete-li předat autorizaci pomocí klíče SSH, musíte si jej vygenerovat nebo najít dříve vytvořený klíč v počítači.

5.1.3 Základní příkazy GitHub

První věc, kterou potřebujeme, je stáhnout a nainstalovat Git do našeho zařízení. Metody závisí na operačním systému našeho počítače. A podrobné pokyny pro každý z nich lze nalézt ve veřejné doméně[10].

Po dokončení všech instalačních aktivit se ujistíme, že se v počítačovém systému zobrazí Git. Otevřeme terminál a zadáme git --version, která by se měla zobrazit aktuální verzí programu v počítači. Tato kontrola je vhodná pro všechny operační systémy.

Poté, co se Git objeví v našem počítači, musíme zadat své údaje, jmenovitě své jméno a e-mailovou adresu. Naše akce v Git budou tyto informace obsahovat. Otevřeme terminál a použijeme následující příkazy:

Tab. 1 Konfigurace nástroje

git config --global user.name "name"	zadejte jméno, pod kterým budou revizí podepsány
git config --global user.email "email"	zadejte e-mailovou adresu, která bude v popisu revizí

Zdroj: vlastní zpracování.

Jen bych upozornily na skutečnost, že výše uvedené příkazy mají možnost --global. „To znamená, že tato data budou uložena pro všechny naše aktivity git a již není nutné je zadávat. Pokud chceme tyto informace měnit pro různé projekty, zadáme do adresáře projektu stejné příkazy, pouze bez možnosti --global.“[39]

Je důležité si uvědomit, že celý proces se provádí prostřednictvím repozitářů, jinými slovy je to pracovní adresář s naším projektem. Ve skutečnosti jde o stejnou

[10] https://git-scm.com/downloads

složku s jakýkoli kódem a dalšími soubory, která je umístěná na serveru GitHub. Takže můžete pracovat s projektem vzdáleně na počítači, a nemusíte obávat, že dojde ke ztrátě souborů – všechna data budou v repozitáři za předpokladu, že jsme je tam poslat.

Tab. 2 Příkazy pro **v**ytvoření repozitáře

git init	vytvořit nový projekt v aktuálním adresáři
git init <folder-name>	vytvořit nový projekt v zadaném adresáři

Zdroj: vlastní zpracování.

Tab. 3 Příkazy pro klonování repozitáře

git clone <repo-link>	klonování vzdáleného repozitáře do stejnojmenného adresáře
git clone <repo-link> <folder-name>	klonování vzdálené repozitáře do adresáře "Název složky"
git clone <repo-link> .	klonování repozitáře do aktuálního adresáře

Zdroj: vlastní zpracování.

Tab. 4 Příkazy pro zobrazit změny

git status	zobrazit stav repozitáře (nové soubory atd.)
git diff	porovnat pracovní adresář a index
git diff index.html	porovnat soubor z pracovního adresáře a indexu
git diff main <second>	podívat se, co se děje ve větvi second ve srovnání s větví main

Zdroj: vlastní zpracování.

Z předchozích kapitol víme, že když pracuje vývojový tým, existuje hlavní verze projektu (main) a jeho větev, se kterou může pracovat jak jedna osoba, tak i celý tým. V terminologii vývojářů se ale velmi často používá výraz fork. Toto je vaše osobní kopie hlavního repozitáře, se kterou můžete dělat, co chcete, a nemusíme se bát, že bychom něco rozbili v hlavní verzi projektu.

Absolutně každý repozitář má alespoň jednu větev. Toto je hlavní větev, kterou Git sám vytvoří, nazývá se main. Pokud chceme přidát do projektu novou funkcionalitu nebo vyzkoušíme nějakou technologii, ale nechceme rozbít kód v hlavní větvi, rozvětvíme z main a pracujeme na své nové větvi. Každá větev je jakousi vedlejší cestou, která se poté napojuje zpět na hlavní silnici.

Tab. 5 Příkazy pro vedení větvení

git branch	zobrazit seznam větví
git branch <branch-name>	vytvořit novou větev s daným názvem
git branch -f main 1754	přesunutí hlavní větve do zadané revize

git checkout <branch-name>	přejít na zadanou větev
git checkout -b new_branch	vytvořit novou větev s daným názvem a přejít na ni
git merge new-branch	přesunout data z větve new_branch do větve, ve které se nacházíme
git branch --merged	zobrazit větve, které již byly spojeny s aktivními
git branch -a	zobrazit všechny existující větve

Zdroj: vlastní zpracování.

Než odešleme všechny změny v souborech a budeme postupovat podle změn, které s nimi byly provedeny, musíme tento soubor přidat do sdílené složky.

Tab. 6 Příkazy pro přidání změn do indexu

git add .	přidat do indexu všechny nové, upravené a odstraněné soubory z aktuálního adresáře a jeho podadresáře
git add file.txt	přidání souboru ''file.txt'' do indexu

Zdroj: vlastní zpracování.

Tab. 7 Příkazy pro zrušení změn

git checkout text.txt	vrátit zpět všechny změny provedené v souboru
git clean -df	odstranění nevystopovatelných souborů a adresářů

Zdroj: vlastní zpracování.

Nyní můžeme provést revizi, to znamená zafixovat všechny uložené změny a pojmenovat je.

Tab. 8 Příkazy pro revize

git commit -m "Name of commit"	potvrzení indexovaných změn (potvrzení), přidání zprávy
git commit -a -m "Name of commit"	indexovat sledované soubory (pouze sledované, ale ne nové soubory) a přidat zprávu

Zdroj: vlastní zpracování.

Když provádíme veškerou požadovanou práci s forkem, musíme vytvořit požadavek na sloučení našeho kódu s daty v hlavním úložišti.

Tab. 9 Příkazy pro požadavek na vyžádání (pull request)/Push

git push	zaslat commit s rychlou kritickou změnou na main ve vzdáleného repozitáře
git push – u origin master	odeslat data z místního úložiště do hlavního (do větve main)
git pull origin	přijmout všechny změny ze vzdáleného repozitáře (všechny větve)

Zdroj: vlastní zpracování.

Samozřejmě existuje spousta takových příkazů a ve skutečnosti není nutné je neustále používat. GitHub má velmi pohodlné rozhraní, které je neuvěřitelně snadné

nastavit a ovládat. Jediná věc, pro kterou je terminál opravdu potřeba, je klonování kódu z repozitáře a odeslání zpět s našimi změnami. Pokud však s konzolí nemůžeme pracovat vůbec, lze všechny změny v souboru provést ručně, ale tato metoda bude trochu zdlouhavá a můžeme udělat chybu, jejíž řešení bude nepříjemné.

5.1.4 Vytvoření repozitáře

Vytvoření a správa repozitáře je jedním z nejdůležitějších bodů, a ve skutečnosti není nejsložitější, ale je třeba dodržovat sekvence. K tomu musíme přímo začít pracovat s webovým rozhraním. Prvním krokem je samotná registrace, která vyžaduje, aby spotřebitel zadal své uživatelské jméno a e-mail. Ihned po potvrzení e-mailu se můžeme pustit do samotné práce. Chceme-li to provést, na hlavní stránce GitHub v pravém horním rohu klikněme na profilovou fotografii a vyhledáme „Your repositories“. Klikneme a vidíme před sebou stránku, kde budou v budoucnu umístěny všechny repozitáře, která vytvoříme. Najdeme zelené tlačítko "Nové", klikneme a přejdeme k samotnému procesu vytváření.

Jediné, co je pro nás v tuto chvíli důležité, je pojmenovat repozitář. V názvech repozitářů GitHub nemůžeme používat mezery. Jako dobrá alternativa je použít pomlčkou „-“ k oddělení slov a čísel. Poté, co si musíme vybrat typ úložiště – pokud chceme, aby úložiště bylo soukromé (přístupné pouze nám nebo lidem, kterým dáváme přístup) nebo veřejné (v tomto případě může naše úložiště najít kdokoli na internetu a sledovat naší práci, klonovat atd.). A také máme možnost automaticky generovat soubory, jako je README, nastavit .gitignore (ignorovat zadané soubory) a nastavit licenci, která nastaví pravidla v našem repozitáře. Poté klikneme na „Create repository“ a okamžitě přejdeme do repozitáře, který jsme vytvořili.
Samotné ovládání může probíhat jak prostřednictvím webového rozhraní, tak pomocí konzoly. Zde se berou v úvahu pouze naše preference. Pokud se mluví o správě prostřednictvím webového rozhraní, pak je to velmi jednoduché. Pomocí tlačítka „Add file“ můžeme importovat libovolný soubor z našeho počítače nebo jej vytvořit ručně přímo zde. Přechodem do souboru a kliknutím na ikonu  , můžeme provádět jakékoli změny v textech a kódu. A hned po provedení změn můžeme přidat svojí vlastní revizi.

Pokud dáváme přednost práci s konzolí, pak je sekvence velmi jednoduchá.

Tab. 10 Postup příkazů pro vytvoření repozitáře

cd <path/to/folder>	přesunete se do adresářů, kde budete ukládat soubory
git init	vytvořit úložiště v tomto adresáři
touch readme.md	vytvoření souboru "readme"
git add readme.md	přidání souboru do indexu
git commit –m „Comment“	vytvořit naši revizi k tomuto souboru
git remote add origin <ssh-clone>	přidejte předem vytvořený prázdný vzdálený repozitář
git push -u origin main	odesíláte data z místního úložiště do vzdáleného (do větve main)

Zdroj: vlastní zpracování.

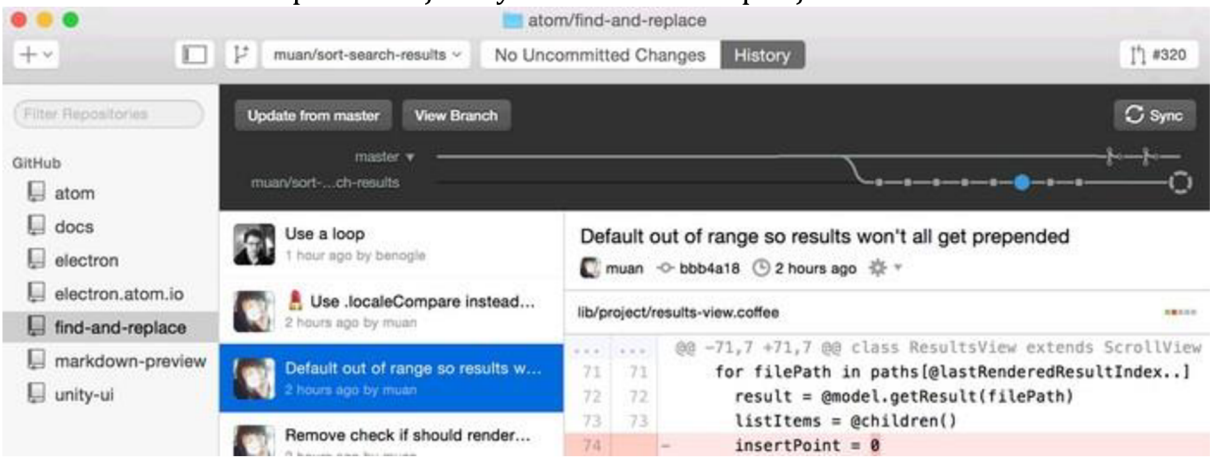
Tímto způsobem GitHub neomezuje a dává nám možnost zvolit si přesně, jak to chceme spravovat.

5.1.5 Rozhraní GitHubu

A i přes přítomnost příjemného a srozumitelného webového rozhraní je pro uživatele vždy zajímavější pracovat v plnohodnotné aplikaci a GitHub není výjimkou. Pokud jde o operační systémy, velmi často dochází ke konfliktům, protože spolupráce, která v sobě mixuje Windows a Mac, nemusí vždy probíhat hladce. Takový konflikt existoval po dlouhou dobu, protože GitHub neměl plnohodnotný nástroj, který by nevytvářel problémy s dohodou mezi dvěma operačními systémy. Zpočátku existovaly pouze Github for Mac a Github for Windows a spolupráce mezi nimi byla opravdu problematická. Proto společnost GitHub financuje vývoj Github Desktop, který má zjednodušit mnoho akcí v pracovním toku a nahradit Github for Mac a Github for Windows novým jednotným rozhraním.

V roce 2015 GitHub oznámil, že vydává nové rozhraní, které bude k dispozici jak uživatelům s bezplatným plánem, tak uživatelům se zvýšenými oprávněními. „Navrhovanou vizí GitHub Desktop je nahradit dříve oddělené aplikace pro Windows a OS X „jednotným rozhraním pro obě platformy“. Grafické rozhraní také obsahuje celý pracovní postup GitHub v jedné aplikaci: od klonování repozitáře, vytvoření větve, fixace změn, vizualizace historie a sdílení kódu.“ [11]

Chceme-li začít pracovat se samotnými repozitáři, můžeme okamžitě vytvořit nový projekt v samotné aplikaci nebo naklonovat existující na GitHub. K tomu stačí jen zkopírovat https odkaz (.git). Je také velmi důležité poznamenat velmi užitečnou a zajímavou inovaci – řádek, který zobrazuje aktuální stav větví a jejich grafické zobrazení. Tímto způsobem je levý horní roh určen pro jeho vizualizaci.



Obr. 11 GitHub Desktop pro OS X

Zdroj: 3dnews.ru

A protože práce v GitHubu je vždy o spolupráci a neustálé interakci mezi vývojáři, aplikace poskytuje velmi pohodlný záznam akcí. V tomto případě, když pracujeme s konkrétním souborem, máme možnost sledovat, kdo s tímto souborem pracoval dříve, jaké konkrétní změny byly provedeny a kdy je tato osoba provedla.

5.1.6 Klávesové zkratky

Pro každého aktivního uživatele jakéhokoli softwaru je přítomnost klávesových zkratek skvělou vlastností, protože klávesové zkratky umožňují rychle provádět různé akce v operačním systému a programech pomocí kombinace kláves bez použití myši a bez vyvolání nabídky akcí. Klávesové zkratky jsou velmi užitečné při provádění často se opakujících akcí při práci v programech. Pokud si zapamatujete jejich kombinace, rychlost uživatelské práce se výrazně zvýší. Protože bude mnohem rychlejší vyvolat nabídku pomocí kombinace několika kláves, než na ni přesunout ukazatel myši, stisknout tlačítko, vybrat požadovanou položku v seznamu a poté znovu kliknout myší. A GitHub není výjimkou.

- Přejít na funkci při kontrole kódu

V případech, kde kontrolujeme více než jednu funkci, je kontrola kódu častým přepínáním mezi voláním funkce a jeho definicí a neustálým posouváním nahoru a dolů. Kliknutím na „t“ zobrazíme vstupní pole, kde můžeme rychle a snadno najít požadovanou funkci.

- Aktivace vyhledávání souborů

  Také klávesa „t“ má mírně odlišné použití, a pokud na něj klikneme při procházení úložiště, zobrazí se řádek pro vyhledávání souborů, do kterého můžeme zadat část cesty a vybrat požadovaný soubor, čímž ušetříme minuty putování kolem úložiště ve vyhledávání.

- Rychlý přechod na konkrétní řádek v souboru

  Při prohlížení souboru stiskneme „l“ - objeví se malé okno pro zadání čísla řádku.

- Rychlé přechody do sekcí GitHub

  Tyto kombinace se snadno dají zapamatovat: začínají předponou „g“ (go, jít), následovanou písmenem označujícím cíl. gp - přejít na seznam požadavků na vyžádání (go pull-request), gi - seznam problémů (go issues), gn - stránka oznámení (go notification) atd.

- Seznam posledních změn

  Při práci se souborem v repozitáře můžeme vidět absolutně všechny změny, které byly s tímto souborem provedeny. Chceme-li to provést, stačí stisknout klávesu „b“ a uvidíme plnohodnotnou tabulku, ve které bude uvedeno, kdy, kým a jaké změny byly provedeny, až po konkrétní řádek kódu, který byl změněn.

Seznam klávesových zkratek tím samozřejmě nekončí a mnoho dalších možností najdeme v oficiální dokumentaci GitHub[11].

5.2 Analýza prostředí GitHub Classroom

Nyní, když jsme prošli základy používání GitHub, bude nejen snazší začít pracovat s nástrojem GitHub Classroom, ale také rychlejší, pokud jde o zpracování informací a

[11] https://docs.github.com/en/github/getting-started-with-github/keyboard-shortcuts

přijetí celého procesu práce. I když je třeba poznamenat, že software GitHub Classroom se stal technologicky vyspělejším a získal nezávislý web, protože na chvíli to byl jen projekt na stránce GitHubu, který bylo potřeba stáhnout a nainstalovat. To bylo způsobeno skutečností, že po určitou dobu byl veškerý vývoj prováděn v otevřeném přístupu, což umožnilo dobrovolníkům zvenčí přispět a propagovat v té době technologicky novou myšlenku. Po chvíli se však vývojáři rozhodli archivovat open source úložiště a pokračovat ve vývoji v rámci společnosti. Tato iniciativa byla argumentována skutečností, že i když taková strategie s otevřeným zdrojovým kódem byla účinná, neposkytla příležitost soustředit se na konkrétní úkoly. Z tohoto důvodu byla kvalita a podpora samotného projektu na relativně nízké úrovni. V důsledku toho bylo učiněno rozhodnutí, které doslova pomohlo GitHubu realizovat všechny jeho možnosti a poskytlo spotřebitelům opravdu vysoce kvalitní produkt.

5.2.1 GitHub Classroom pro učitele

Než přistoupíme přímo k analýze mechaniky práce s GitHub Classroom je důležité podotknout, že tento nástroj byl původně koncipován jako něco, co by pomohlo učitelům na prvním místě. Samotná mechanika nástroje byla postavena na použití GitHub API k povolení pracovního toku GitHub pro vzdělávací účely. A samozřejmě se tento nástroj spoléhal spíše na exaktní vědy – vyšší biologii, statistiku a programování. Jelikož se jedná o stálou práci, která vyžaduje provádění pravidelných změn a interakcí s daty.

Pokud jde o vzdělávací proces, tyto oblasti často vyžadují čas a přesné výpočty – jak bude probíhat kurz, informativnost úkolů pro studenty a jejich dostupnost, ukládání a ověřování stejných úkolů, a to vše s přihlédnutím k tomu, že tam je více než jeden kurz. Právě pro takové potřeby byl navržen nástroj GitHub Classroom, ale když mluvíme o tak rozsáhlé a nové funkčnosti pro průměrného člověka, musíme si pamatovat, že by měl být celý proces práce co nejjasnější a nejpřístupnější. Když si společnost uvědomila riziko, GitHub pokračoval ve vývoji a také poskytl uživatelům nejen rozsáhlou dokumentaci, ale i podporu, kde nám vždy pomohou s našimi otázkami a pokud souhlasíme s plnou spoluprací, staneme se součástí online kurzů efektivního využívání všech funkcí GitHub Classroom.

V této situaci vyvstává další otázka, jak se liší učitel „samouk“ od učitele, který obdrží balíček služeb od GitHub Classroom? Jistě, chceme-li normálně používat nástroj GitHub Classroom, nemusíme nutně žádat o privilegium učitele. Takový balíček má však mnoho výhod. A především je to GitHub Teacher Toolbox[12], který obsahuje kolekci mnoha nástrojů nejen od GitHubu, ale i od jiných sponzorů. Těmito sponzory jsou univerzity s technickým zaměřením, které prokázaly svou osobní iniciativu v oblasti rozvoje. Při vytváření „předplatného“, máme možnost získat přístup k poměrně velké databázi nástrojů (cloudy pro úložiště, domény, různá grafická rozhraní atd.). Kromě toho nám poskytují bezplatné předplatné GitHub Team pro naše kurzy a osobní nekomerční průzkum. Tento tarif zahrnuje 2 GB pro uložení kódu, 3000 minut měsíčně na použití akce GitHub Action, možnost provádět revize atd. Kromě toho je příležitost zapojit se do vzdělávacího kurzu GitHub Campus Advisor, který nám pomůže zvládnout základy práce s Git a GitHub.

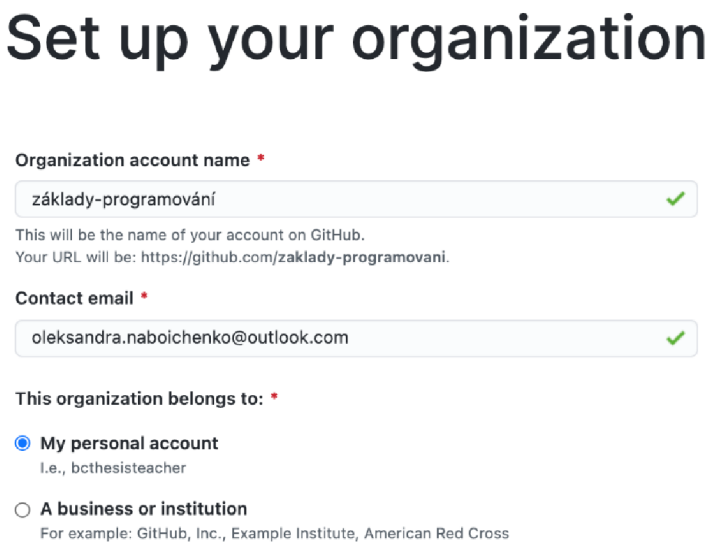
5.2.1.1 Jak začít pracovat s GitHub Classroom

Nyní, když jsme prošli několik klíčových bodů, můžeme začít zkoumat GitHub Classroom očima instruktora. Prvními kroky jsou samozřejmě registrace a nastavení profilu, protože jej nebudeme používat jako normální uživatel, ale přímo učitel vysoké školy. Potom, co přejdeme na hlavní stránku GitHub, musíme projít standardní registrací bezplatného profilu. Údaje, která budeme potřebovat, jsou uživatelské jméno, e-mail a heslo. Po vytvoření účtu můžeme okamžitě začít pracovat s GitHub Classroom, takže stačí přejít na jejich web, přihlásit se a první věc, kterou před sebou uvidíme, je tlačítko "Create your first classroom". Abychom však mohli pokračovat ve vytváření třídy, musíme vytvořit organizaci, kde bude uložen celý seznam všech našich tříd. Od této chvíle je velmi důležité správně naplánovat organizaci kurzů. Vzhledem k tomu, že celý další proces bude zahrnovat správné označení kurzu. To je důvod, proč musíme všechno správně pojmenovat, abychom tomuto konceptu plně porozuměli a v budoucnu nevytvářeli zmatek jak pro učitele, tak pro studenty.

---

[12] https://education.github.com/toolbox

Pokud tedy provádíme kurz základů programování, pak název "basics-of-programming" bude docela vhodný, na druhou stranu nejsme omezeni diakritikou, proto bude vhodný i český jazyk. Poté označíme e-mail, který bude považován za hlavní, a bude dostávat oznámení o globálních změnách, a poslední věc, kterou musíme zadat, je typ organizace.



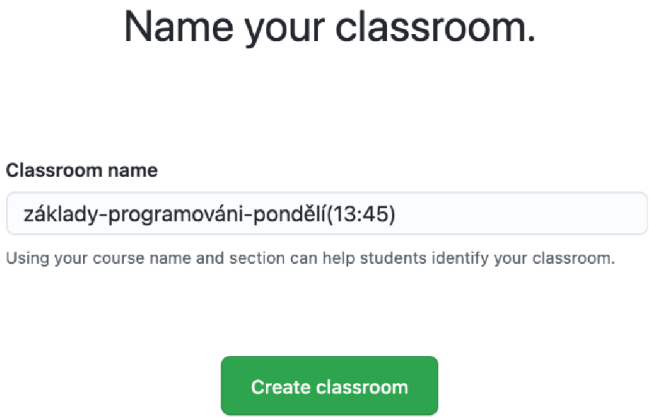
Obr. 12 Nastavení organizace na GitHubu

Zdroj: vlastní obrázek

V našem případě bude nejlepší volbou „My personal account“, protože veškerou správu přebíráme bez jakýchkoli dohod atd. Pokud zvolíme " A business or institution", pak se okamžitě pod tlačítkem pro vytvoření organizace zobrazí upozornění na dohodu s jinými podmínkami použití, které budou obsahovat další podmínky, například že zástupce společnosti může převzít kontrolu organizace atd. Tuto otázku bude muset řešit vzdělávací instituce a také za ni nést zodpovědnost. Proto si prostě vybíráme "My personal account". V budoucnu budeme tuto organizaci používat k ukládání všech tříd a nezbytných dokumentů a úkolů, které budou používány ve studiu tohoto kurzu.

V souladu s tím bude každá úloha v tomto kurzu vytvořena jako samostatný repozitář. Samozřejmě je možné vytvořit jeden repozitář a seřadit úkoly podle složek, avšak s dalším využitím tříd bude mnohem pohodlnější poslat studentům samostatný repozitář, který bude symbolizovat jeden úkol. Tím se zabrání nejasnostem a zbaví nás možnosti psaní obrovském návodu k použití. A samozřejmě bude velmi snadné provádět změny v budoucnu.

Nyní, když byla vytvořena naše organizace, můžeme začít organizovat samotný kurz. Po vyplnění formuláře pro naši organizaci a stisknutí tlačítka „Next“ přejdeme na stránku s organizacemi a vybereme tu, kterou jsme právě vytvořili. Okamžitě se ocitneme ve formuláři pro vytvoření učebny, kde vygenerujeme standardní název pro naši třídu. Nicméně, není nutné bezmyšlenkovitě stisknout tlačítko pokračovat, protože je velmi snadné tím udělat zmatek. Je velmi důležité vytvořit určitý druh specifičnosti pro název třídy. Ať je to číslo, nebo den a čas, kdy se kurz koná.

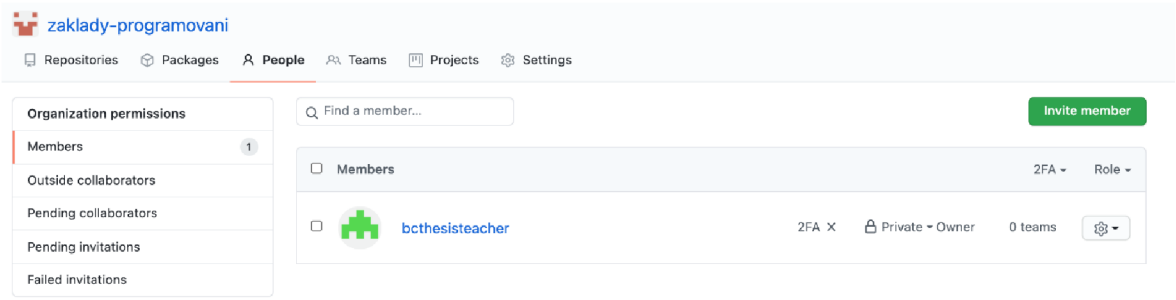


Obr. 13 Nastavení učebny na GitHubu Classroom

Zdroj: vlastní obrázek

Dalším bodem, co nám GitHub Classroom nabízí, je přidat možné asistenty a administrátory třídy. V tomto případě může být administrátorem jakýkoli učitel nebo technický specialista, který stejně jako vy bude mít kontrolu nad třídami v GitHub Classroom a schopnost pomáhat s kontrolou úkolů od studentů. Chceme-li přidat svého kolegu do svého celkového vývoje kurzu, je prvním krokem přidat ho jako zaměstnance naší organizace na GitHub. Proto přejdeme na hlavní stránku GitHubu a prostřednictvím nabídky, která se otevře, když klikneme na fotografii profilu v pravém horním rohu, klikneme na "Your organizations", přesuneme se do

organizace, kterou jsme právě vytvořili (klikneme na její název) a vybereme sekci "People".



Obr. 14 Přidávání asistentů a správců tříd v GitHubu

Zdroj: vlastní obrázek

Kliknutím na tlačítko „Invite member“ zadáme uživatelské jméno nebo e-mail, prostřednictvím kterého je náš kolega registrován na GitHubu, a označíme, že ho chceme přidat jako vlastníka. V takovém případě si bude moci kurz nejen prohlédnout, ale také jej upravit a aktivně se účastnit vývoje. Poté náš budoucí asistent obdrží e-mail, prostřednictvím kterého se stane spolupracovníkem v naší organizaci na plný úvazek. Chceme-li však začít pracovat na samotném kurzu, budeme se muset vrátit do GitHub Classroom, vybrat naši učebnu a přejít do sekce „TAs and Admins“, kde je třeba zkopírovat odkaz na naši třídu a sdílet ji s kolegy. Po tomto kroku se všichni kolegové, které přidáme, stanou plnohodnotnými spolupracovníky a jsou připraveni učinit první kroky.

5.2.1.2 Připojení systému pro správu učení k GitHub Classroom

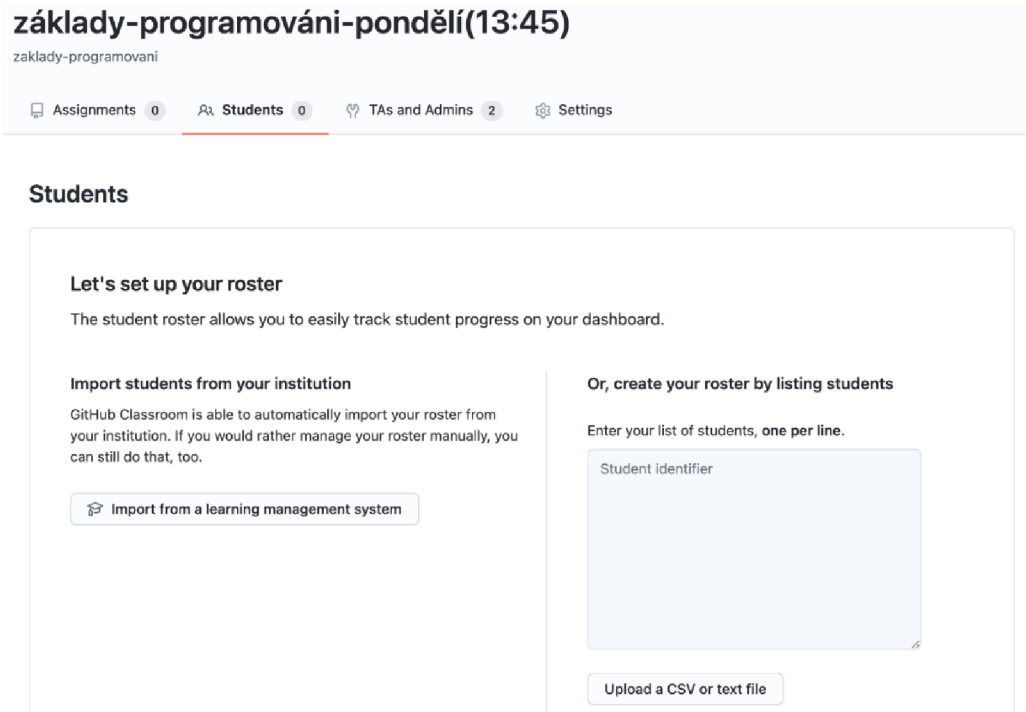
Jak již bylo zmíněno vícekrát, změna standardního vzdělávacího systému a jeho principů přináší řadu problémů. Protože učitelé jsou zvyklí na metody a platformy, se kterými tak dlouho pracují, a zavádění něčeho nového není jen obtížný proces obecně, ale také stres. To je důvod, proč GitHub Classroom velmi rychle začal přemýšlet o tom, jak usnadnit práci těm, kteří jsou zvyklí na známá rozhraní. Kromě obvyklého manuálního přidávání studentů nám tedy GitHub Classroom umožňuje připojení prostřednictvím našeho systémů řízení učení (Learning Management System, LMS) a import seznamů studentů.

Postupně s rozvojem e-learningu byla naléhavá potřeba sledovat externí zdroje, které studenti používají ke zlepšení svých pracovních zkušeností. Tímto způsobem IMS Global Consorsium, které se již dlouho zabývá rozvojem technologií v oblasti

vzdělávání, vyvinulo komunikační prostředky, které pomohly propojit nástroje třetích stran se systémy řízení učení – Learning Tools Interoperability (LTI). Učitelé tedy nemusí ručně importovat seznamy studentů z LMS do GitHub Classroom ručně, ale import dat provádí právě díky této technologii.

Abychom si zjednodušili celý tento proces, GitHub umožňuje konfigurovat LTI, aby byla implementace pohodlnější, a nevyžadovala další práci. V tuto chvíli však GitHub Classroom nepodporuje práci s platformami, jako jsou Blackboard a Brightspace, ale vzhledem k tomu, jak populární je GitHub a jeho nástroje, doufáme, že se v blízké budoucnosti objeví nová modifikace.

Chceme-li připojit svůj LMS, musíme na webu GitHub Classroom přejít na požadovanou třídu a přejít do sekce „Students“. A právě zde stojíme před volbou, jak chceme studenty přidat.



Obr. 15 Přidávání studentů do tříd

Zdroj: vlastní obrázek

Pokud zvolíme možnost „Import from a learning managment system“, zobrazí se zde seznam možných LMS, které lze v tuto chvíli konfigurovat. Zatím je možné importovat data pouze z následujícího LMS:

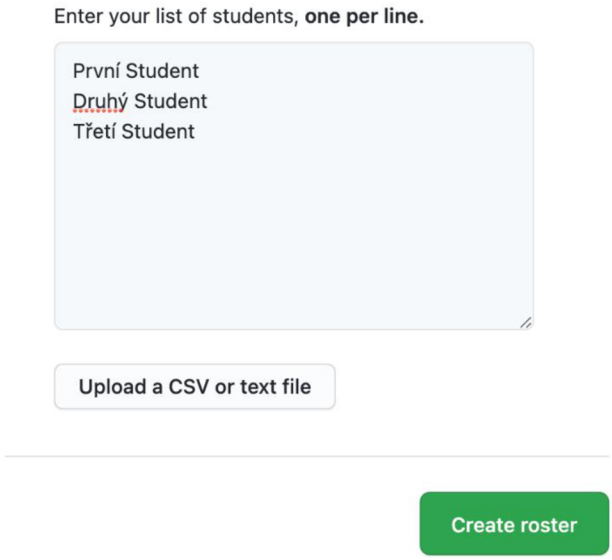
- Google Classroom
- Canvas

- Moodle
- Sankai

Jakmile si vybereme platformu, kterou potřebujeme, obdržíme „Consumer Key“, „Shared Secret“ a „Launch URL“, které budeme muset použít pro připojení k našemu LMS. Obvykle se od nás vyžaduje, abychom navštívili náš LMS, nastavili externí nástroj a zadali konfigurační pověření uvedená výše. Také v oficiální dokumentaci GitHubu najdeme podrobnou konfiguraci pro platformy, jako jsou Canvas a Moodle[13].

5.2.1.3 Ruční výpis studentů

Pokud naše vzdělávací instituce nepoužívá systém řízení učení, je také možné sestavit seznamy buď ručně na samotném webu nebo jej sestavit ve formátu .csv nebo v jiném textovém dokumentu. Jediná věc, kterou je důležité kontrolovat, je samozřejmě jméno a příjmení, které lze považovat za jedinečné, ale nezapomeňme, že existují případy, kdy se setkají lidé se stejným jménem a příjmením. V takovém případě můžeme použít studentův e-mail nebo jeho ISIC kód.

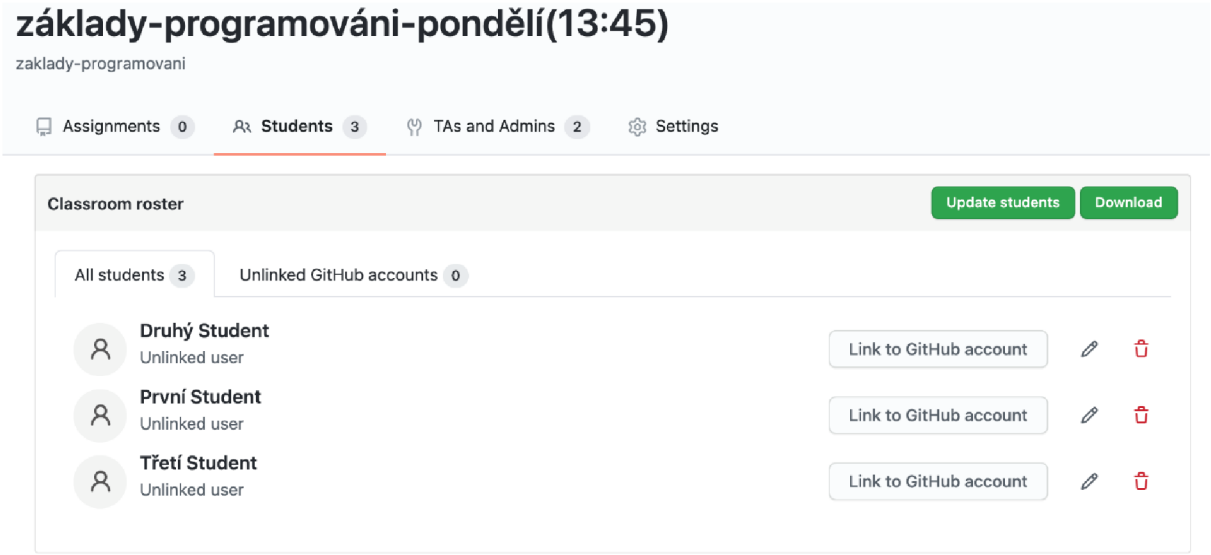


Obr. 16 Příklad sestavení seznamu studentů

Zdroj: vlastní obrázek

[13] https://docs.github.com/en/education/manage-coursework-with-github-classroom/connect-a-learning-management-system-to-github-classroom#about-configuration-of-your-lms

Ihned poté, co zadáme všechny studenty nebo importujeme textový dokument, klikneme na „Create roster“. Po přidání seznamu to bude vypadat takto (obr. 18) v tomto případě uvidíme pouze data, která jsme zadali, ale samotní studenti nebudou připojeni k účtu GitHub. Budou však připojeni, jakmile vytvoříte první domácí úkol.



Obr. 17 Vytvořený seznam studentů, kteří ještě nejsou připojeni ke kurzu

Zdroj: vlastní obrázek

Právě tam máme příležitost provádět drobné manipulace se seznamem. První věc, která upoutá pozornost, je „Update students“, zde můžeme přidávat nové studenty stejným způsobem, jako jsme je přidali dříve. Můžeme tu upravit jméno, pokud bylo zadáno nesprávně, nebo odstranit uživatele ze seznamu, pokud byl přidán omylem. Poté si můžeme seznam stáhnout, ale dokud nebudou studenti připojeni prostřednictvím svého profilu GitHub, bude stahování seznamu téměř prázdné a bude obsahovat pouze jméno a příjmení, které jsme zadali.

5.2.1.4 Vytváření individuálních úloh

V této fázi „vývoje“ můžeme říci, že jsme na cestě k nejzajímavější části, protože nás čeká úkol naplnit náš kurz. V této chvíli musíme mít určitý základ, jak budeme kurz vést a co budeme od svých studentů požadovat. Hlavní otázkou, nad kterou se budeme muset zamyslet je, jaký bude formát projektů, buď individuální nebo skupinový.

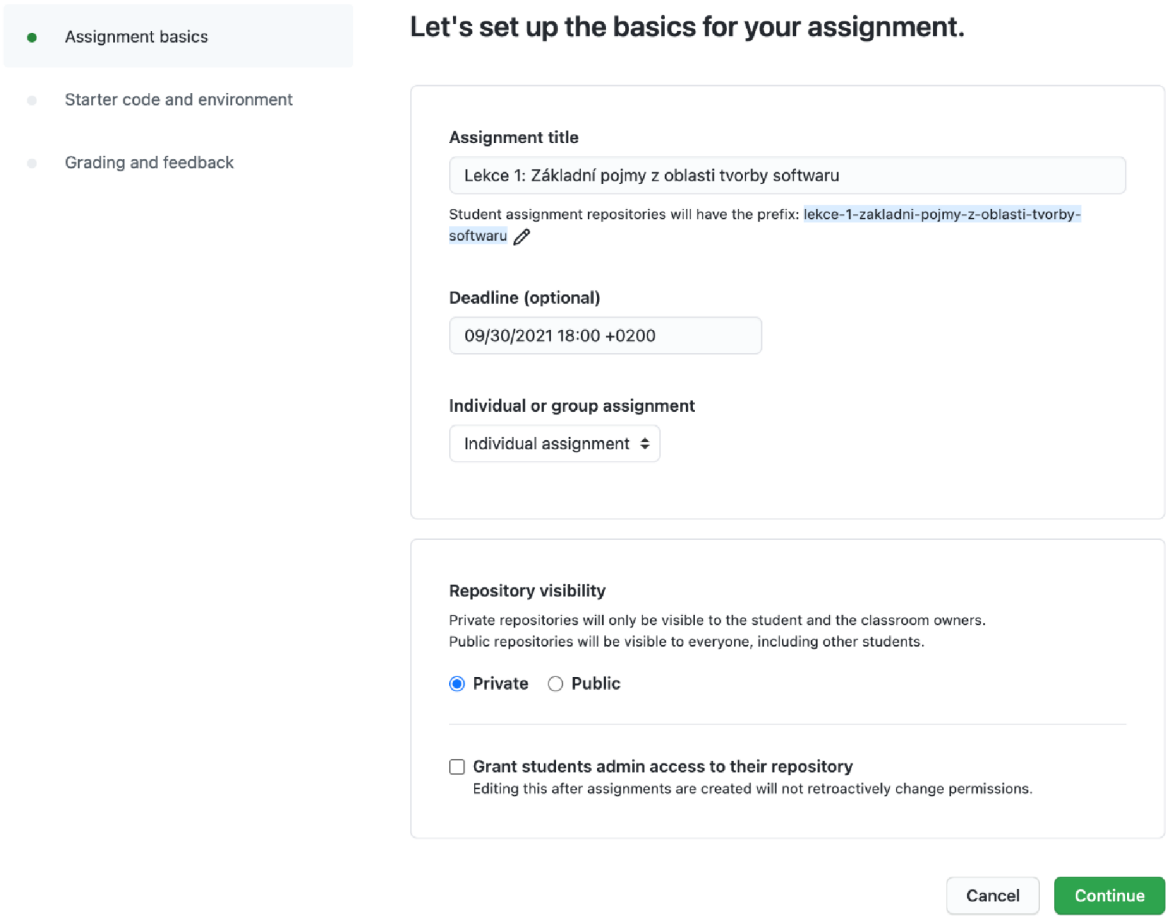
Koncept individuálního úkolu je velmi snadný - je to úkol, který je určen pro každého studenta individuálně a neposkytuje vzájemnou práci a plagiátorství v rámci kurzu. Tímto způsobem můžeme vytvořit jak samostatný úkol pro každého studenta, nebo zvolit obecné téma a přidat k němu některá podtémata, pro která úkol každého studenta získá individualitu. V GitHub Classroom práce s takovým úkolem vypadá takto, student úkol přijme, poté GitHub Classroom pro studenta automaticky vytvoří repozitář, ve kterém bude moci pracovat a rozvíjet proces plnění. V tomto případě je náplň tohoto repozitáře výhradně na nás, můžeme si vytvořit plnohodnotný repozitář, který bude obsahovat šablonu pro návrh nebo počáteční kód, praktické příklady, které mohou být užitečné, dokumentaci a další informace, které považujeme za vhodné. Na druhou stranu může zůstat úložiště prázdné, a informovat studenty můžeme jakékoliv jiným způsobem.

Tak či onak, každý úkol obsahuje nadpis a máme také možnost stanovit termín pro odeslání projektu, ale tato položka není nutná. Včetně jako hlavní manažer můžeme spravovat viditelnost repozitáře, a tedy pouze my rozhodujeme, kdy budou studenti moci získat přístup k určité složce.

Abychom vytvořili individuální úkol, jako vždy se přesuneme do třídy, kterou chceme, a poté zkontrolujeme, zda se nacházíme v sekci „Assignments“. V době psaní této práce GitHub Classroom poskytuje dvě možnosti, vytvořit si vlastní úkol nebo použít testovací verze od GitHub, který můžeme použít k výuce studentů základům Git a GitHub. Abychom však mohli úplně projít všemi fázemi přípravy úkolu, musíme si jej vytvořit sami. Proto klikneme na tlačítko „Create an assignment“ a přesuneme se k vyplnění formuláře.

Vyplníme název našeho úkolu, zohledníme naše preference, protože neexistuje přesná šablona pro vyplňování. Poté můžeme nastavit termín, který studenti uvidí, jakmile se pustí do vyplňování. Zvolíme typ našeho úkolu - „Individuální“. Máme také možnost nastavit viditelnost repozitářů, díky čemuž se určí, zda ostatní studenti budou mít možnost vidět repozitář s kódem ostatních studentů, nebo ne. Pokud tedy zřídíme „private“ úložiště – studenti ho vidět nemohou, při „public“ - mohou. A posledním nastavením je schopnost poskytnout studentům status správce z repozitáře. Pokud tedy student chce konfigurovat nástroje CI nebo provést jakékoli jiné nastavení, bude potřebovat status správce, ale

takové nastavení je žádoucí pouze v kurzu, kde jsme si jisti dovednostmi a znalostmi studentů, jak používat GitHub. Pokud mluvíme o nováčcích, riskujeme, že se stavem správce mohou poškodit strukturu projektu, takže toto pole ponecháme nezaškrtnuté. A klikněme na „Continue“.



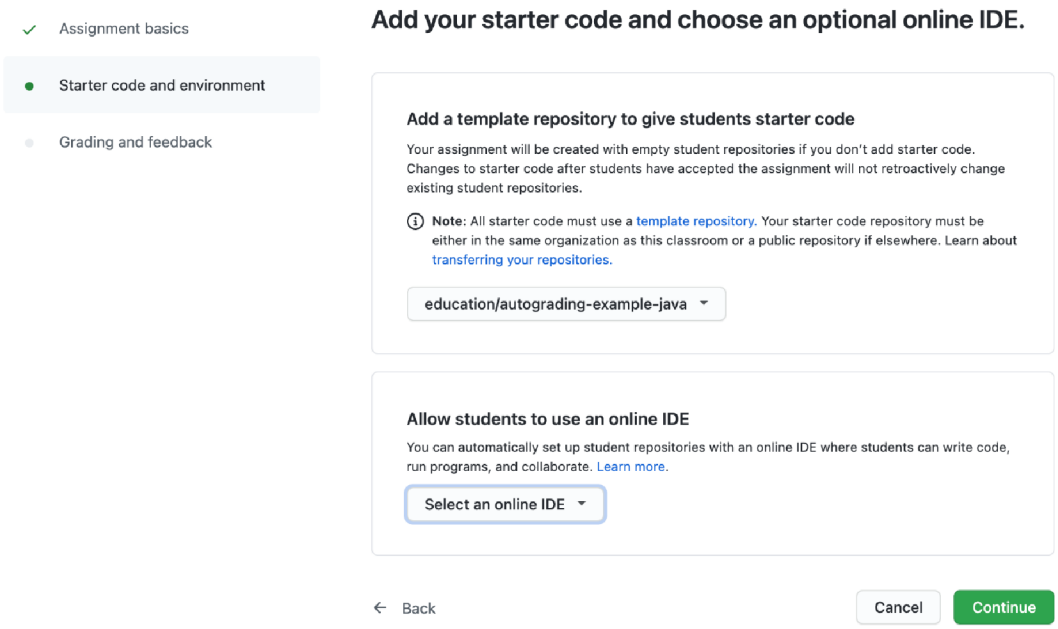
Obr. 18 Hlavní fáze pro vytvoření úkolu

Zdroj: vlastní obrázek

V této fázi musíme připojit repozitář s úkolem a pokyny k němu. Zpočátku se předpokládá, že si předem vytvoříme repozitář, který bude vyhovovat našim osobním požadavkům. Například pokud chceme studentům poskytnout praktické příklady bez jejich přiřazení nebo naopak, vytvoříme obecný úkol s nějakou dokumentací a podpůrnými materiály. Tady nás nikdo neomezuje. Je však třeba si uvědomit, že pokud chceme importovat vytvořený repozitář do naší třídy, musí být nutně veřejný, i když chceme importovat náš vlastní repozitář. Bude také mnohem pohodlnější, když bude nakonfigurován jako „template“. Z libovolného úložiště lze vytvořit šablonu, za tímto účelem je nutné po jeho vytvoření přejít na něj, najít sekce

“Settings” a zaškrtnout „Template repository“. Když to uděláme, repozitář se stane šablonou, díky níž můžeme my a další uživatelé rychle vytvořit nové repozitáře se stejnou strukturou adresářů, větví a souborů. Mohou také zahrnovat všechny ostatní pobočky ve vašem repozitáři.

Tímto způsobem můžeme také importovat veřejné repozitáře jiných uživatelů. V tomto případě použijeme veřejný repozitář, který nám poskytuje GitHub Classroom[14]. Chceme-li importovat náš první repozitář, jednoduše klikneme na „Select a repository“ a poté musíme ukázat, který konkrétní repozitář chceme specifikovat. Abychom to mohli udělat, musíme pro nalezení úložiště dodržovat velmi jednoduchou strukturu GitHubu, a to „uživatelské-jméno/název-repozitáře“. Poté ve spodním panelu vybereme repozitář, který potřebujeme, a klikneme na něj.



Obr. 19 Připojení repozitáře k úloze

Zdroj: vlastní obrázek

Než přejdeme k dalšímu kroku, je třeba zmínit jeden bod. GitHub Education velmi pečlivě sleduje nejen uživatelské recenze, ale také nejnovější trendy v oblasti vzdělávání. Takže pro vytváření a práci s úlohami, GitHub Classroom poskytuje možnost implementovat platformy třetích stran. První možností, která se obvykle považuje za klasickou, je vytvoření úkolu přímo v GitHubu, studenti si jej stáhnou a

[14] https://github.com/education/autograding-example-java

splní zadání pomocí jakékoli vhodné aplikace (Eclips, Visual Studio Code, IntelliJ IDEA atd.). Další navrhovaná možnost již používá online platformu nazvanou Microsoft MakeCode[15]. V tomto případě musíme vytvořit úlohu právě na této platformě a poté importovat všechny potřebné soubory na GitHub jako repozitář. Tento proces je však omezen jediným stisknutím tlačítka. Poslední možností je použít server Repl.it[16], který nám umožní psát a kompilovat kód přímo v prohlížeči. Ale protože se cíl naší práce zaměřuje na práci se samotným GitHub, přeskočíme podrobný úvod k těmto platformám a budeme pokračovat s klasickou variantou. Proto v sekci „Allow students to use an online IDE“ nic nevybereme, protože ve výchozím nastavení není online server specifikován. Ale klikneme na pokračovat.

Přesunuli jsme se do poslední fáze nastavení našeho úkolu, a to přidání automatické kontroly a možnosti zpětné vazby pomocí požadavku na vyžádání. V tomto případě automatická kontrola znamená, že pokaždé, když student odešle svůj dokončený úkol, server automaticky spustí test, který jsme nakonfigurovali pomocí GitHub Actions. Po těchto testech budeme mít svůj vlastní seznam známek a student bude mít příležitost několikrát přepracovat a odeslat kód, aniž by vyžadoval naše osobní ověření. Obecně se testy dělí na dva typy:

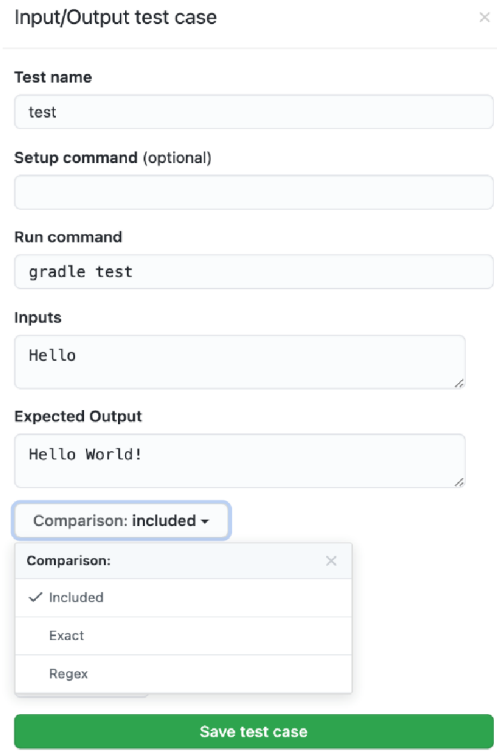
- Input/Output

  Toto je druh manuálního nastavení testu. Hlavním bodem pro nastavení takového testu je poskytnout příkaz, který zahájí celý proces testování, poté test nechá uvést Input, který jsme zadali, a počká na odpověď z programu napsaného studentem. Budeme mít také možnost zvolit, jak přesně bude test porovnávat odpovědi - Included, Exact, Regex.

  Included bude úspěšné, pokud se očekávaný výsledek objeví kdekoli v odpovědi na spuštěný test; Exact bude úspěšná, pouze pokud je odpověď

[15] https://www.microsoft.com/en-us/makecode/about
[16] https://replit.com/

totožná s očekávaným výsledkem; Regex uspěje, pokud přijatý výstup obsahuje nějaké shody se zadanou správnou odpovědí.



Obr. 20 Příklad vytvoření Input/Output testu

Zdroj: vlastní obrázek

- Run command test

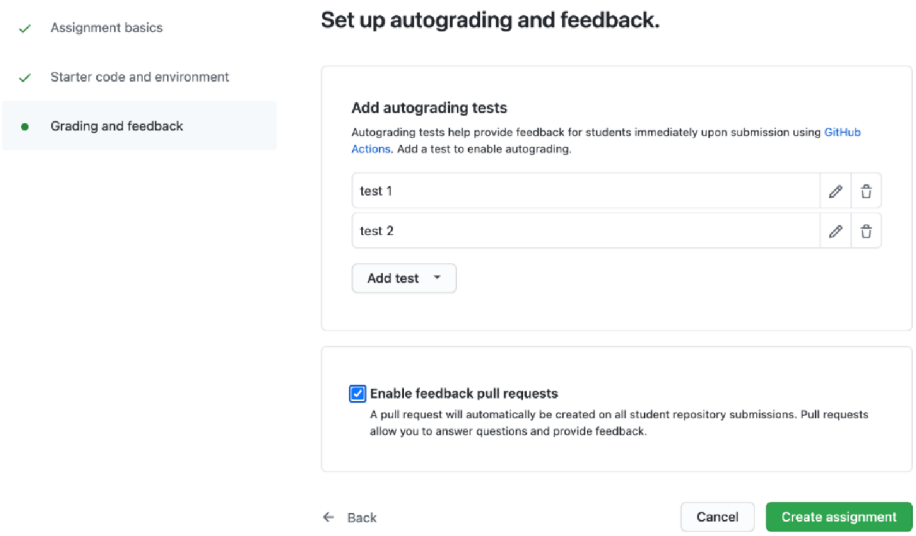
  Vyžaduje od nás minimální přizpůsobení, ale budeme muset předem napsat testovací kód. V takovém případě, jakmile student odešle svůj kód, automatická kontrola spustí příkaz k provedení testu, po kterém se očekává 0, což znamená, že test byl úspěšný. Pokud obdržíme jinou odpověď než nulu, znamená to, že výstup kódu studenta neodpovídá kritériím, které požadujeme pro test, který jsme napsali. Vezměme si například test, který nám poskytuje GitHub Education, v tomto případě by měl být výsledek „Hello world!“, pokud je však výstup studenta jiný, test selže.

```
Hello.main(null);

// assertion
assertEquals("Hello world!\n", bos.toString());

// undo the binding in System
System.setOut(originalOut);
```

Obr. 21 Testovací kód z GitHub Education

Zdroj: github.com

Poslední věcí, kterou bych chtěla poznamenat je zpětná vazba, která proběhne hned, jak student odešle projekt. Při odesílání, bude mít student příležitost zanechat dotaz nebo požádat o radu, která bude připojena ke kódu, proto v takovém případě budeme mít před očima studentovu práci, ve které můžeme najít chybu nebo navrhnout, jak vylepšit kód. Tato funkce však bude aktivní, pouze pokud povolíme tento typ komunikace.



Obr. 22 Nastavení testů a zpětné vazby

Zdroj: vlastní obrázek

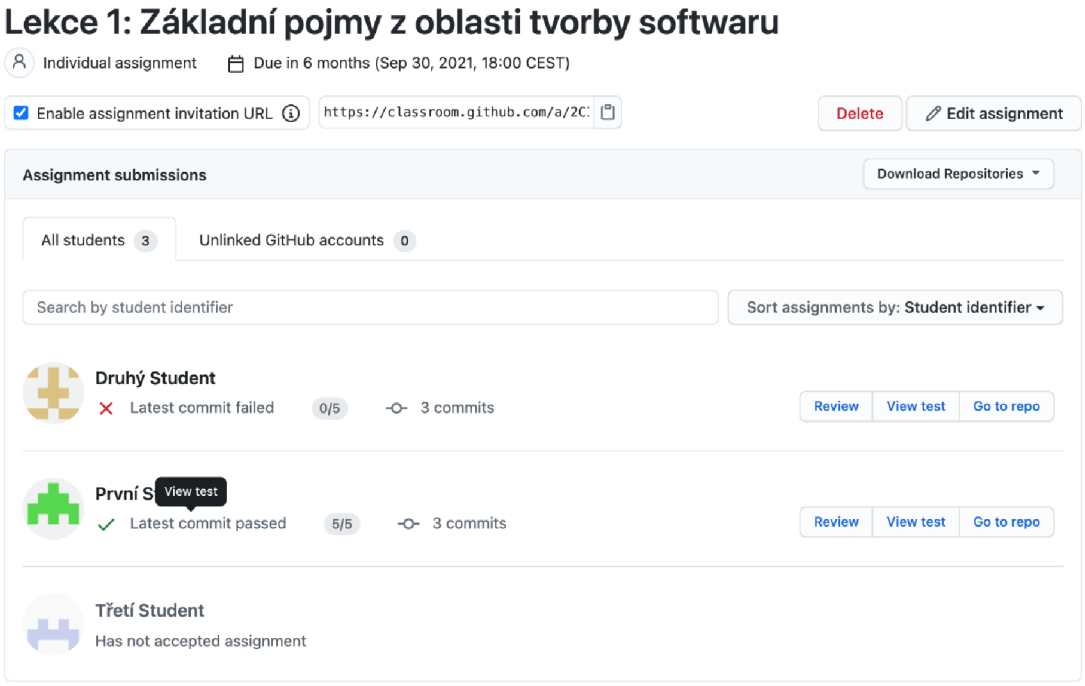
Po kliknutí na tlačítko „Create assignment“ se vytvoří náš úkol a poslední věc, kterou od nás vyžaduje, je poslat automaticky vygenerovaný odkaz na úkol naším studentům. Proto jej pouze zkopírujeme a pošleme studentům.

Obr. 23 Aktivní odkaz na vytvořený úkol

Zdroj: vlastní obrázek

Také vedle odkazu můžeme vidět checkbox, který dělá odkaz aktivní. Pokud tedy odstraníme zaškrtnutí a student se ještě nestihl zapojit do úkolu, nebude se moci dostat do naší třídy, dokud ji znovu neaktivujeme.

Ihned poté můžeme pokračovat ve vytváření dalších úkolů. Nebo pokračovat v řízení již existujících. Například pokud přejdeme na stránku svého úkolu, okamžitě uvidíme, který ze studentů se již k úkolu připojil a jestli už má díky automatickým testům známku. Kliknutím na tlačítko „Review“ naproti jménu studenta přejdeme do požadavků na vyžádání konkrétního studenta a budeme moci vstoupit do konverzace, pokud máme nějaké dotazy, nebo naopak zanechat posouzení. Podle tohoto můžeme sledovat celkovou aktivitu studentů, jak prošli testy, zjistit jaké změny byly provedeny a kdy, strukturu větví atd.



Obr. 24 Vzhled rozhraní úlohy

Zdroj: vlastní obrázek

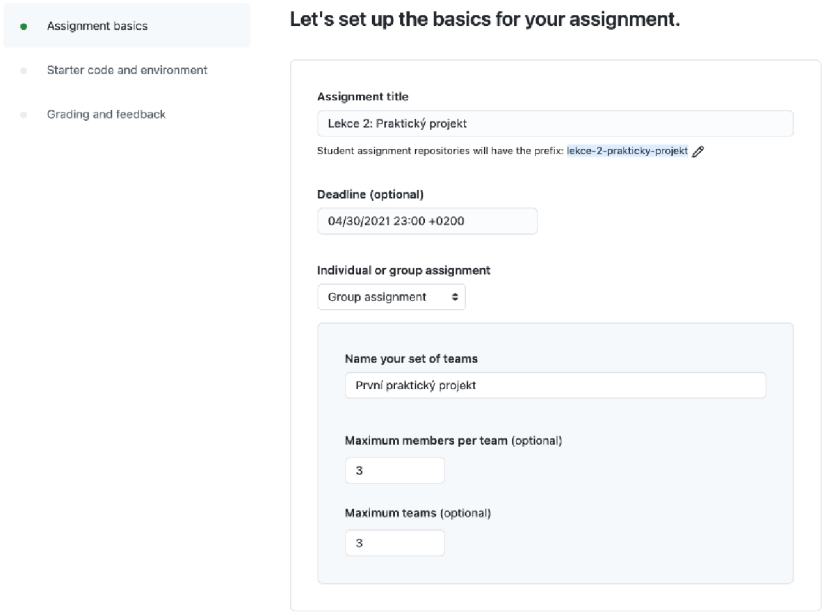
5.2.1.5 Vytváření skupinových úloh

A samozřejmě skupinové úkoly, na vysokých školách jsou takové projekty povinné, protože učit studenty pracovat jako tým je jednou z nejdůležitějších dovedností. GitHub Classroom se tedy snaží podporovat rozvoj takové důležité dovednosti, zejména na jejich platformě.

Zaprvé, když studenti přijmou skupinové úkoly, mají příležitost jak si vytvořit vlastní osobní skupinu, a tak se také připojit ke stávající. Pokud tedy nejde o první skupinový projekt, GitHub Classroom uloží týmy, které již byly formulovány, a v případě několika týmových prací budeme mít možnost vybrat již formulované skupiny studentů.

Stejně jako u individuálních repozitářů se při práci v týmu vytvoří jeden repozitář na skupinu a podle toho, pokud se v kódu provedou nějaké změny, bude historie projektu označovat, kdo a kdy něco změnil. Pro učitele je tedy velmi snadné posoudit individuální zapojení každého ze členů skupiny.

Nyní se podívejme na to, jak se tvorba individuálního úkolu liší od skupinového. Prvních pár kroků je naprosto identických. Přejdeme do GitHub Classroom, vybereme třídu, kterou potřebujeme, klikneme na „New assignment“, pojmenujeme náš úkol a nastavíme termín. Dále musíme naznačit, že chceme vytvořit skupinový úkol, a poté přijde několik změn. Pokud se jedná o náš první projekt zaměřený na společnou práci, budeme muset uvést název, pod kterým budou v budoucnu uloženy všechny formulované skupiny.



Obr. 25 Nastavení skupinového úkolu

Zdroj: vlastní obrázek

A při vytváření dalšího skupinového úkolu nebudou muset být naši studenti znovu rozděleni do skupin, ale stačí zadat název skupiny, který jsme uvedli poprvé,

a již budeme mít složení týmu. Bude také možné nakonfigurovat maximální počet týmů a kolik maximálně lidí může být v jednom týmu.

Všechny ostatní postupy zůstávají nezměněny, ale stojí za zmínku, že funkčnost soukromého repozitáře se nemění, a proto k němu budou mít přístup absolutně všichni členové týmu. Proto dokončíme všechna požadovaná pole a zašleme odkaz úkol našim studentům.

5.2.2 GitHub Classroom pro studenty

Princip práce pro studenty se příliš neliší od běžné práce v GitHub. V době, kdy se učitelé potřebují seznámit s novým rozhraním, novými funkcemi a obecně existuje spousta nových informací, které je třeba zvládnout, studenti budou mít dostatek znalostí základů. Ale i díky tomu, že v dnešní době probíhá výměna informací s neuvěřitelnou rychlostí, je naprosto snadné najít podporu nebo řešení konkrétního problému. Koneckonců, na jakýkoli náš problém s pravděpodobností 99% existuje otevřená diskuse s řešením.

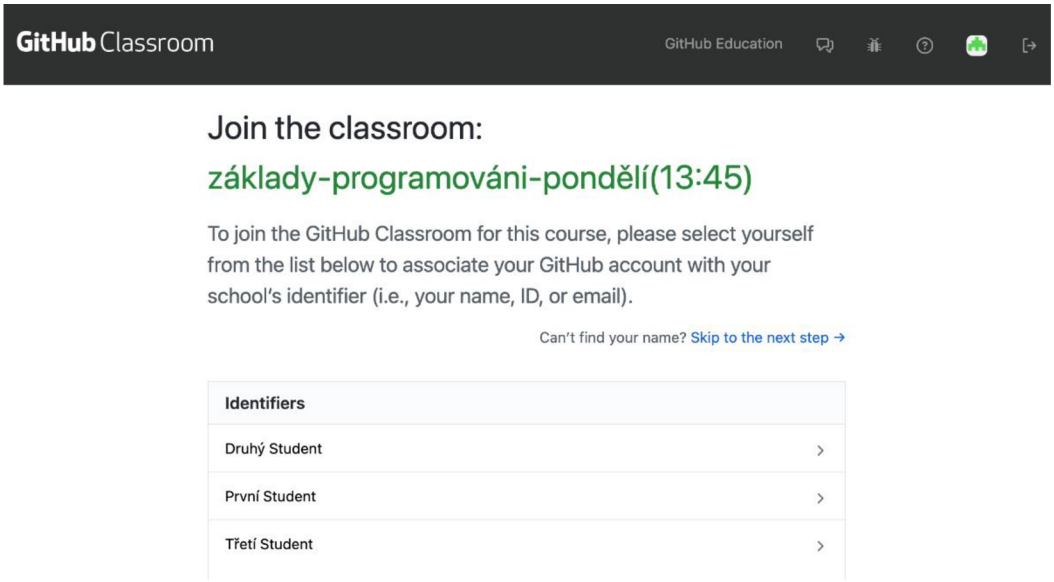
Co se týče studentských privilegií, zde je o čem diskutovat. Nejprve studenti nezískají obvyklý bezplatný profil GitHub, ale GitHub Pro, který poskytuje e-mailovou podporu, více minut pro použíti Github Actions, více nástrojů a úložiště. Každý student také obdrží GitHub Student Developer Pack[17]. Stejně jako u učitelů se jedná o bezplatný přístup k nástrojům od partnerů, které máme k dispozici, pokud jsme studentem. Poslední možností je příležitost stát se součástí GitHub Campus Expert. Existují také dvě možnosti studovat a stát se jedním z lídrů v naší vzdělávací instituci nebo najít takového člověka, který nás bude informovat o nových trendech, obdrží mistrovské kurzy a zúčastní se hackathonů.

5.2.2.1 Přijetí individuálního úkolu

Když student obdrží odkaz na repozitář, posloupnost akcí je velmi jednoduchá. Pokud student poprvé otevře odkaz od učitele, pak prvním krokem, který bude muset udělat, je druh registrace. Jakmile otevřeme odkaz, který nám byl zaslán, uvidíme před sebou seznam všech studentů přidaných učitelem. První věc, kterou

[17] https://education.github.com/pack

musíme udělat, je najít své údaje v seznamu studentů a okamžitě kontaktovat učitele, pokud jsme v seznamu nenašli svoje údaje nebo jsme vůbec neobdrželi odkaz. Jakmile najdeme své jméno v seznamu, klikneme a potvrdíme, že se opravdu chceme přihlásit jako tento konkrétní student. Tímto způsobem připojíme svůj účet GitHub k této třídě pod námi zvoleným jménem.



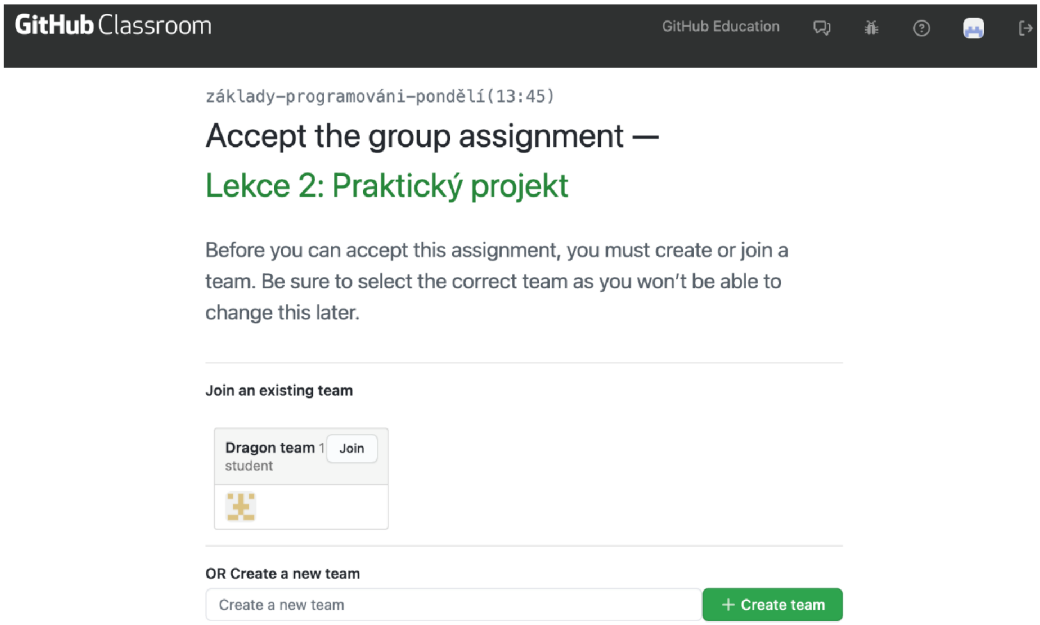
Obr. 26 První připojení studenta ke třídě

Zdroj: vlastní obrázek

První věcí, kterou je nutně poznamenat je, že pro každého studenta bude vygenerováno individuální úložiště, které bude obsahovat název úkolu a jméno studenta (uživatelské jméno GitHub). Co se týče osobního repozitáře studenta, stojí za zmínku několik bodů, za prvé, repozitář studenta nijak neovlivní hlavní repozitář a za druhé, veškeré změny, které budou provedeny (odstranění kódu, psaní komentářů, dokumentace atd.) budou zaznamenány nejen pro samotného studenta, ale také pro učitele.

5.2.2.2 Přijetí skupinových úloh

Jako v případě jednotlivých úkolů musíme počkat na odkaz, protože nemáme jiný způsob, jak se zapojit a vytvořit si vlastní tým. Když otevřeme odkaz, okamžitě uvidíme týmy, které již byly vytvořeny se všemi členy, a možnost připojit se k jakémukoli nebo vytvořit vlastní osobní skupinu. V závislosti na výběru dostaneme buď již vygenerovaný repozitář pro existující tým, nebo bude poskytnut nový repozitář pro naši skupinu.



Obr. 27 První připojení studenta do skupinového úkolu

Zdroj: vlastní obrázek

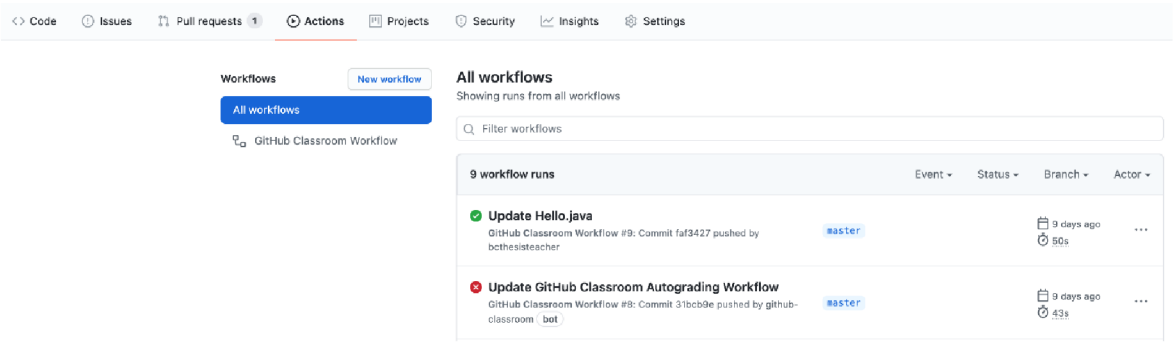
Ihned poté se staneme součástí skupiny na plný úvazek a můžeme začít pracovat sami, nebo se okamžitě dohodnout se svými partnery na další práci. Každý, kdo s námi na této práci pracuje, bude uveden v pravém sloupci „Contributors“. Pokud jde o samotnou spolupráci, pro zahájení diskusí můžeme použít sekci „Issues“, kde otevřeme konkrétní otázku a volně diskutujeme se zbytkem studentů formou komentářů. Další komunikace je možná také prostřednictvím požadavků na vyžádání, ale pouze v případě, že učitel povolí tento typ komunikace. Nelze však vyloučit ani jiné možné způsoby komunikace, ať jde o sociální síť nebo e-mail.

5.2.2.3 Prohlížení výsledků automatického hodnocení

Obvykle, když mluvíme o odevzdání projektu, chápeme, že máme pouze jeden pokus o odevzdávání, protože pro učitele je kontrola projektu velkou spotřebou osobního času a zdrojů. V tomto případě by neměli studenti udělat sebemenší chybu, protože to může vést ke špatnému hodnocení. V GitHub Classroom nepotřebujeme posílat projekt učiteli k vyhodnocení, stačí nám poslat projekt do našeho repozitáře a za pár

minut obdržíme výsledek automatického hodnocení. Aby to však bylo možné, musí náš instruktor nastavit možnost takového hodnocení.

Abychom mohli zkontrolovat naše výsledky, stačí přejít do našeho repozitáře v GitHubu a otevřít složku „Actions“. V něm okamžitě uvidíme výsledky všech testů, které byly provedeny po celou dobu existence repozitáře. Pokud jsou testy úspěšné, zobrazí se zelená značka zaškrtnutí. Pokud testy selžou, bude vidět červený křížek. Chceme-li zobrazit podrobné informace o testu, můžeme kliknout na zelené zaškrtnutí nebo na červený křížek.



Obr. 28 Příklad úspěšného a neúspěšného testu

Zdroj: Vlastní zpracování

6 Výzkumy dotazníkového šetření

Dotazník byl zaměřen především na obecné posouzení využití moderních technologií, jako je GitHub Classroom, na vysokých a středních školách. Proto byli dotázáni studenti z celé České Republiky. Tím bylo možné rozšířit spektrum dotazovaných a neomezovat se pouze na univerzity s technickým zaměřením.

První část dotazníku byla zaměřena na pochopení toho, jak v současné době probíhá komunikace mezi studenty a učiteli, jak studenti dostávají informace o projektech, a také na zjištění osobního hodnocení studentů, jak učitelé hodnotí praktické úkoly studentů. Druhá část dotazníku byla o tom, jak studenti sami pracují s projekty a jak jim v tom vzdělávací instituce pomáhají. Jinými slovy, otázky byly zaměřeny na to, jak studenti ukládají projekty, jak fungují skupinové projekty a zda jim instituce poskytují možnost ukládat projekty na školní zdroje. Posledních pár otázek již byly zaměřeny na to, zda dotazovaní studenti vědí, co je VCS, jaké produkty této technologie znají a jak se o nich dozvěděli.

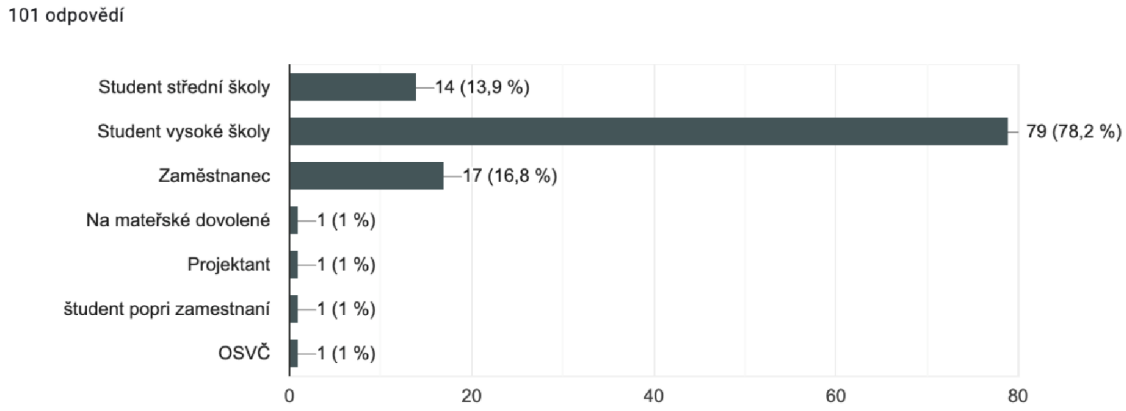
Samotný dotazník se prováděl elektronicky prostřednictvím Formuláře Google. Dotazník byl anonymní a počítal se studenty různých věkových kategorií. Samotný dotazník se skládá z 21 otázek, z toho 4 otázky jsou nepovinné a 17 je povinných. Dotazník byl šířen prostřednictvím sociálních sítí do univerzitních skupin. V době psaní tohoto článku prošlo dotazníkem 101 lidí.(viz. Příloha 1)

6.1 Vyhodnocení výzkumu

Otázka č. 1: Jste:

Tato otázka měla na výběr více než jednu odpověď a možnost přidat vlastní odpověď. Na základě statistiky otázky dostaneme následující výsledky - 1 osoba (0,9 %) zvolila tři možnosti současně, a to Student vysoké školy, Zaměstnanec, Na mateřské dovolené; 11 lidí (10,9 %) odpovědělo – Student vysoké školy a Zaměstnanec; a také 1 osoba (0,9 %) si vybrala - Student vysoké školy a OSVČ. Zbytek zvolil jednu odpověď: 13,9 % - Student střední školy; 66,9 % - Student vysoké školy; 5,9 % - Zaměstnanec; 0,9 % - projektant.

Ačkoliv byl samotný dotazník zaměřen pouze na studenty, takový výsledek nebude mít na výsledek žádný vliv, protože se předpokládá, že všichni dotazovaní prošli nebo procházejí fází vzdělávání.



Obr. 29 Graf, aktuální stav respondentů

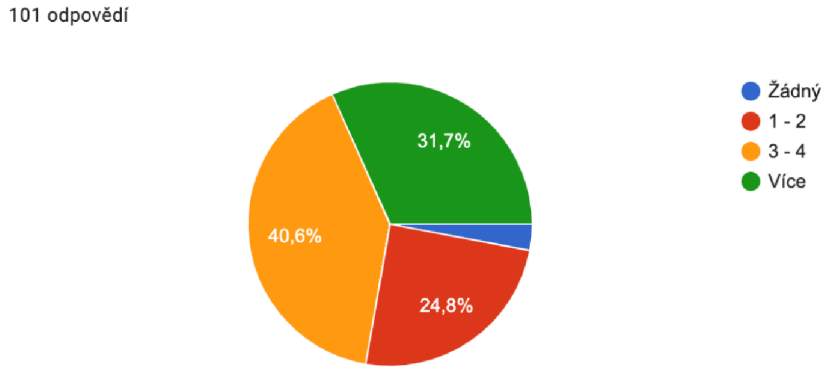
Zdroj: Vlastní zpracování

Otázka č. 2: Jaký obor studujete?

Tato otázka byla nastavena s otevřenou odpovědí, aby respondenti mohli specifikovat, v jaké oblasti studují, a protože otázka byla povinná, rozsah odpovědí je velmi odlišný.

Při sestavování dotazníku nebylo hlavním cílem omezit se na lidi s jednou konkrétní specializací. A přesně to se stalo, protože spektrum studentů je velmi odlišné, od filologů po exaktní vědy.

Otázka č. 3: Kolik můžete mít za jeden semestr seminárních prací? (projekty v textových dokumentech, praktické projekty jakéhokoli typu)

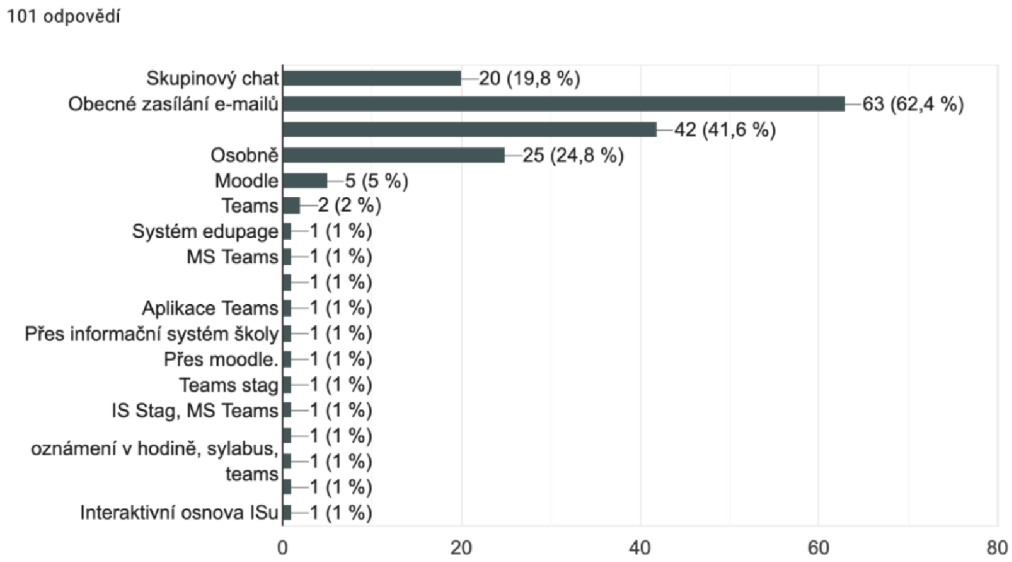


Obr. 30 Graf, počet projektů za semestr

Zdroj: Vlastní zpracování

Nyní pomalu přecházíme k hodnocení samotného procesu učení. Tato otázka ukazuje, že téměř ve všech profesích je alespoň jeden projekt za semestr a pouze 3 (3 %) studenti neměli ani jeden projektový úkol. Zbytek 97 % má alespoň jeden projekt během jednoho semestru, bez ohledu na to, zda se jedná o skupinový nebo individuální úkol. Ale většina studentů má výrazně více než jeden projekt, protože pouze 24,8 % má 1 až 2 projekty, 40,6 % má 3 až 4 projekty za semestr a 31,7 % má více než 4 projekty. Díky tomu jasně chápeme, že samostatná práce a sebevzdělávání se ve vzdělávacích institucích aktivně praktikuje. Protože každý praktický úkol vyžaduje, aby student věnoval více čas studiu.

Otázka č. 4: Jakým způsobem získáváte osobní úkoly od učitelů?



Obr. 31 Graf, použité aplikace pro získávání úkolů

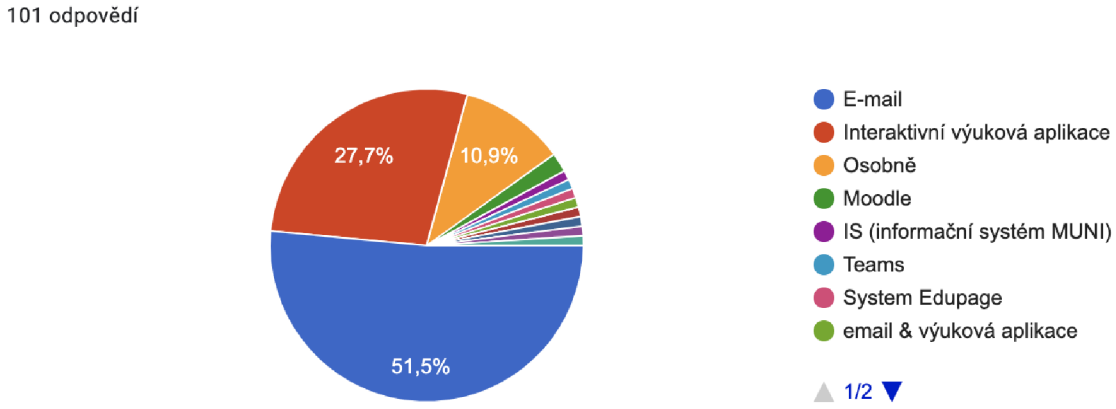
Zdroj: Vlastní zpracování

Tato otázka měla také na výběr více než jednu odpověď. V tomto grafu jasně vidíme, jak vzdělávací instituce implementují aplikace pro interaktivní výuku a další populární servery pro rychlou a pohodlnou komunikaci mezi studenty a učiteli.

8 lidí (7,9 %) si jako jedinou možnost na tuto otázku vybralo "Osobně" a 17 (16,8 %) také jako jednu odpověď zvolilo "Obecné zasílání e-mailů". Zbývajících 75,3 % studentů zaznamenalo více než jednu odpověď, vytvořilo různé kombinace a přidalo také platformy s možným okamžitým připojením, jako jsou Microsoft Teams a Moodle.

Samozřejmě, tato statistika je velmi dobrá, protože ukazuje touhu vzdělávacích institucí zavádět novější metody a platformy. A to je velmi důležité, i když vezmeme-li v úvahu podmínky pandemie, jsou pro výměnu informací k dispozici pouze online servery. Na druhou stranu, když vezmeme v úvahu zbývajících 24,7 %, zde jsou udržovány poměrně zastaralé metody výměny materiálů.

Otázka č. 5: Jak získáváte zpětnou vazbu od učitele?



Obr. 32 Graf, způsoby zpětné vazby

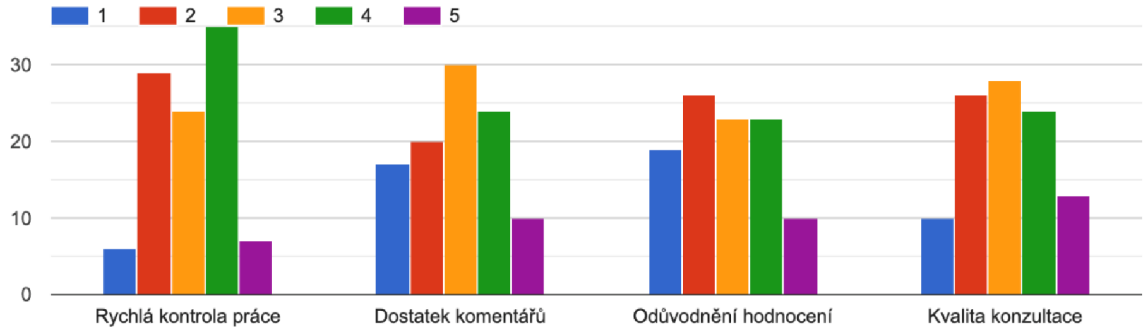
Zdroj: Vlastní zpracování

A samozřejmě přímá komunikace mezi studentem a učitelem. Jedna věc je posílat studentům materiály a úkoly, a úplně jiná udržovat komunikaci a zůstat v kontaktu s každým studentem. V tomto případě výsledek ukazuje, že většina stále dodržuje komunikaci prostřednictvím e-mailu. Podle toho tedy 51,5 % všech respondentů komunikuje pouze prostřednictvím e-mailu, 27,7 % volí aplikace pro interaktivní výuku, a 11 osob (10,9 %) uvedlo komunikaci "Osobně". Zbývajících 9,9 % si vybralo vlastní možnost, která buď kombinovala několik možností, nebo označily platformy s okamžitou komunikací.

Otázka č. 6: Jste spokojeni s kvalitou zpětné vazby (1 - nejnižší hodnocení, 5 - nejvyšší hodnocení)?

Tato otázka byla zaměřena na zjištění přímo od studentů, jak jsou spokojeni s metodami výměny informací a komunikace, které zvolili učitelé. Podle toho je 1 nejnižší skóre a 5 nejvyšší. A na základě grafů je obtížné jednoznačně posoudit

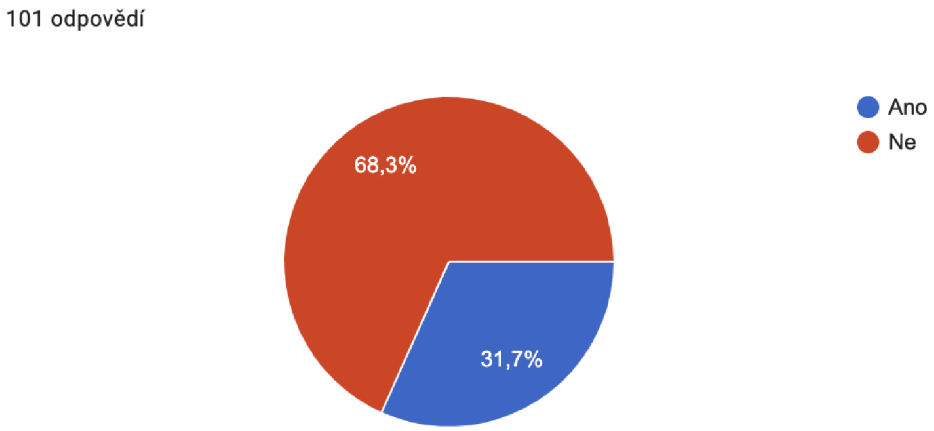
výsledek, protože odpovědi jsou udržovány na průměrné úrovni. Chtěla bych však poznamenat, že „Odůvodnění hodnocení“ a „Dostatek komentářů“ shromáždily největší počet nízkých známek. A také graf „Rychlá kontrola práce“, který je také pozoruhodný, ukazuje většinou pozitivní recenze, a to 34,9 %, kteří dosáhli 4 bodů, ale stejných 28,9 % studentů, kteří dosáhli 2 bodů, což znamená, že se čekací doba na ověření zpožďuje.



Obr. 33 Graf, hodnocení zpětné vazby

Zdroj: Vlastní zpracování

<u>Otázka č. 7: Stalo se vám, že učitel vaši práci nehodnotil (zapomněl/nemohl najít), nebo ji ztratil?</u>



Obr. 34 Graf, chybějící hodnocení projektu

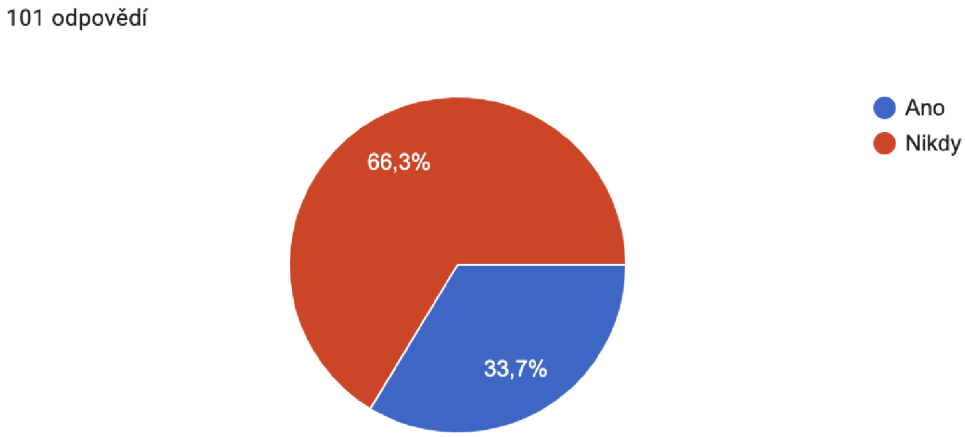
Zdroj: Vlastní zpracování

Díky tomuto grafu máme možnost vidět, že 31,7 % studentů mělo zkušenost s tím, že učitel nechal studenta bez známky na projektu. A můžeme poznamenat, že se jedná o poměrně velké procento, taková situace sama o sobě nemusí být

nejpříjemnější, protože může narušit rozvrh výuky studenta. Na druhou stranu máme 68,3 %, kdy taková situace nenastala a kontrola projektů proběhla úspěšně.

Hlavní motivací pro otázku bylo posoudit organizaci učitelů. Protože pokud jde o použití aplikace pro interaktivní výuku, pravděpodobnost, že zapomenete zkontrolovat nebo ztratit projekt je extrémně malá, protože již existují seznamy. Pokud je však úkol odeslán a přijat prostřednictvím e-mailu, otázka organizace spadá přímo na učitele. Což může vést k takovým následkům.

Otázka č. 8: Stalo se vám, že byl soubor s projektem poškozen v průběhu práce, v důsledku čehož data nebyla uložena nebo byla zcela ztracena?



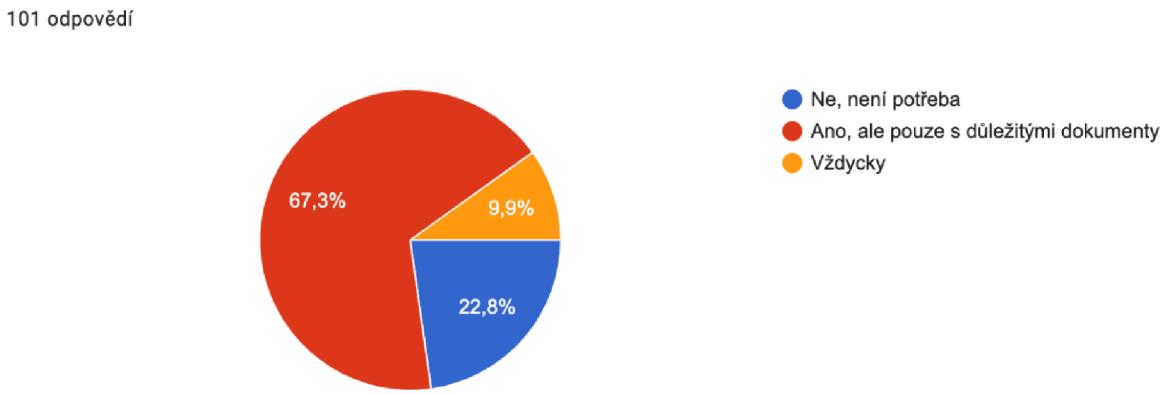
Obr. 35 Graf, poškozených projektu

Zdroj: Vlastní zpracování

Další pár otázek již bude zaměřeno na hodnocení organizace studentů. V éře technologií se skutečně otevírají nové možnosti ukládání, ale také se vytváří možnosti nízkého rizika a ztráty dokumentů.

Na základě grafu, tedy 33,7 % studentů mělo nepříjemné případy ztráty nebo poškození dokumentu, 66,3 % se však nikdy nesetkalo s takovým problémem.

Otázka č. 9: Máte ve zvyku multi-ukládání (ukládání důležitých dokumentů různými způsoby = ukládání do počítače, cloudu a na sociální sítě současně)?

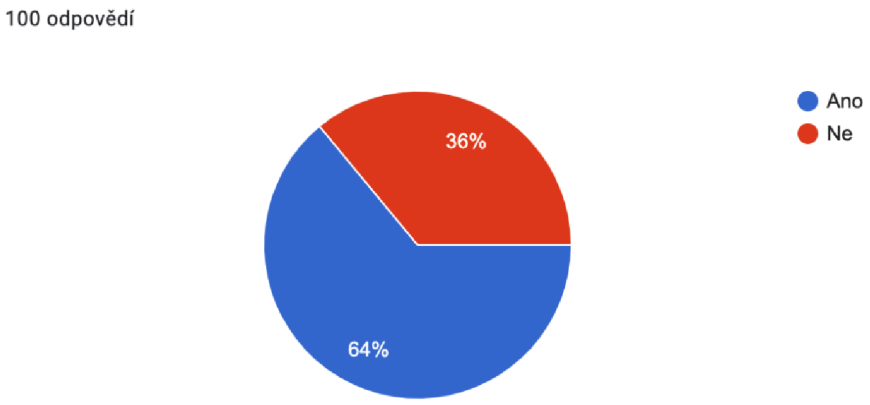


Obr. 36 Graf, multi-ukládání

Zdroj: Vlastní zpracování

Z tohoto grafu můžeme vyvodit, že většina studentů dává přednost ukládání důležitých dokumentů různými metodami (ukládání do počítače, cloudu atd.). Pouze 9,9 % studentů vždy ukládá soubory pomocí různých metod, 67,3% dává přednost ukládání pouze důležitých dokumentů a 22,8 % nevidí potřebu takové metody ukládání.

Otázka č. 10: Poskytuje vaše škola možnost ukládat vaše materiály na školní zdroje (místo na školním pevném disku, cloud, školní server)?



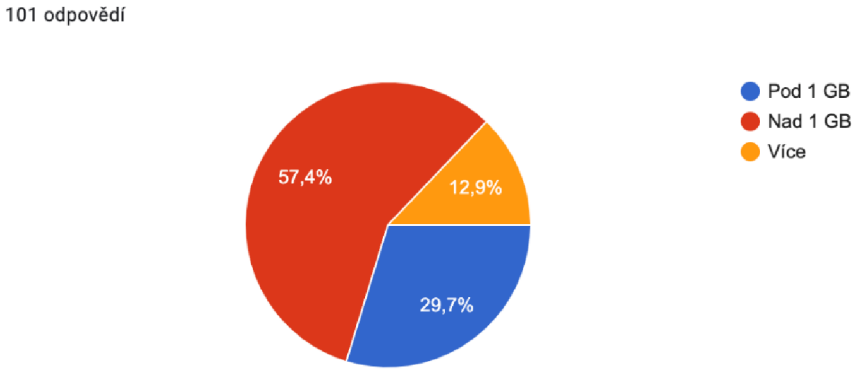
Obr. 37 Graf, možnost ukládat na školní zdroje

Zdroj: Vlastní zpracování

Tento graf částečně demonstruje technologický pokrok škol, protože poskytnout studentům schopnost ukládat osobní dokumenty na zdroje samotné vzdělávací

instituce je docela nákladné podnikání. Ve vzdělávacích institucích má tedy 64 % studentů možnost ukládat jakékoli soubory, pokud jsou studenty v této škole, na druhou stranu 36 % studentů jsou zbaveni této možnosti.

Otázka č. 11: Kolik paměti (přibližně) potřebujete k uložení školních materiálů?

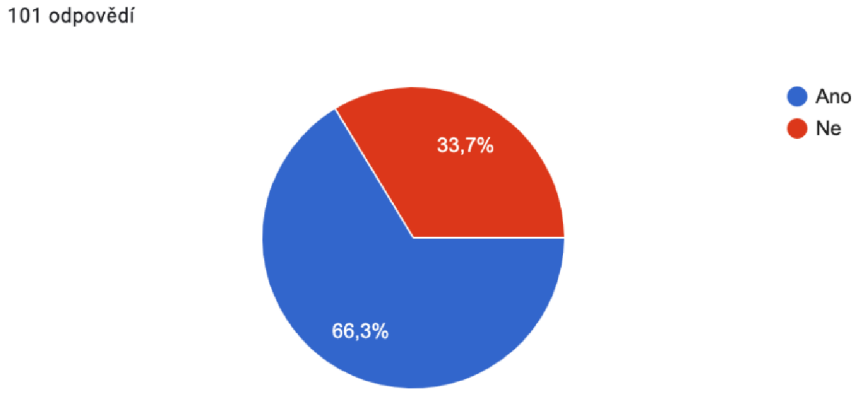


Obr. 38 Graf, využití paměti pro ukládání školních materiálů

Zdroj: Vlastní zpracování

V této otázce se měly za cíl dozvědět, kolik průměrně studenti potřebují prostoru pro uložení studentských materiálů. Koneckonců, v průběhu celého vzdělávacího procesu mohou studenti mít velké množství projektů a potřebují ukládat učebnice v elektronické podobě atd. Podle grafu vyplývá, že 29,7 % studentů má studentské materiály menší než 1 GB, 57,4 % studentů má více než 1 GB a pouze 12,9 % potřebuje paměť přesahující 2 GB.

Otázka č. 12: Potřebujete pracovat na skupinových projektech?

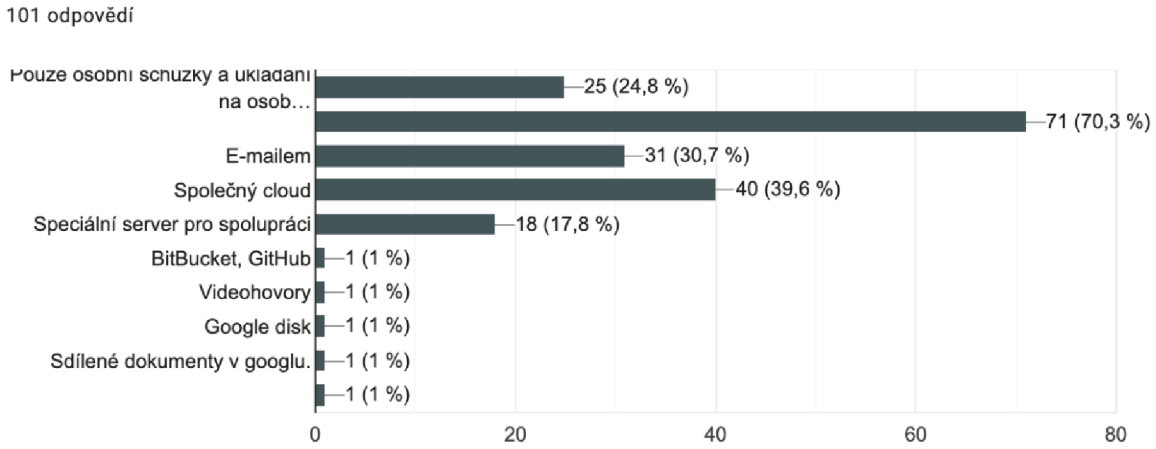


Obr. 39 Graf, skupinové projekty

Zdroj: Vlastní zpracování

Řada dalších otázek se zaměřila na skupinové projekty a na to, jak studenti organizují svou týmovou práci. Díky tomuto grafu vidíme, že 66,3 % studentů má skupinové projekty a 33,7 % je nemá vůbec.

Otázka č. 13: V případě skupinového projektu jakým způsobem preferujete výměnu provedených změn v souboru?

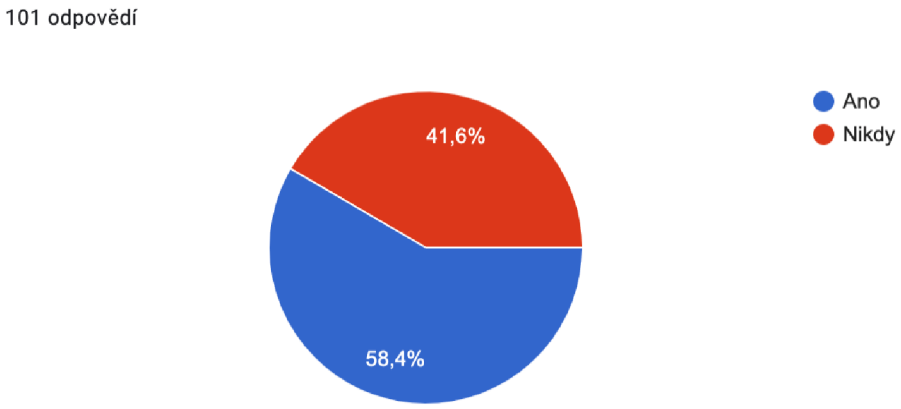


Obr. 40 Graf, způsob výměny změn ve skupinovém projektu

Zdroj: Vlastní zpracování

Tato otázka měla na výběr více než jednu odpověď a možnost přidat vlastní. A téměř všechny odpovědi byly kombinací různých možností, což je již vynikající ukazatel. 70,3 % studentů volí výměnu informací přesně „Prostřednictvím sociálních sítí“, 39,6 % vytváří „Společný cloud“ pro společnou práci, 30,7 % volí e-mail, 24,8 % preferuje osobní schůzky a ukládání projektů pouze na osobním počítači a pouze 17,8 % volí možnost „Speciální server pro spolupráci“ například webhosting.

Otázka č. 14: Museli jste někdy přepracovat projekt kvůli k tomu, že poslední provedené změny byly chybné?



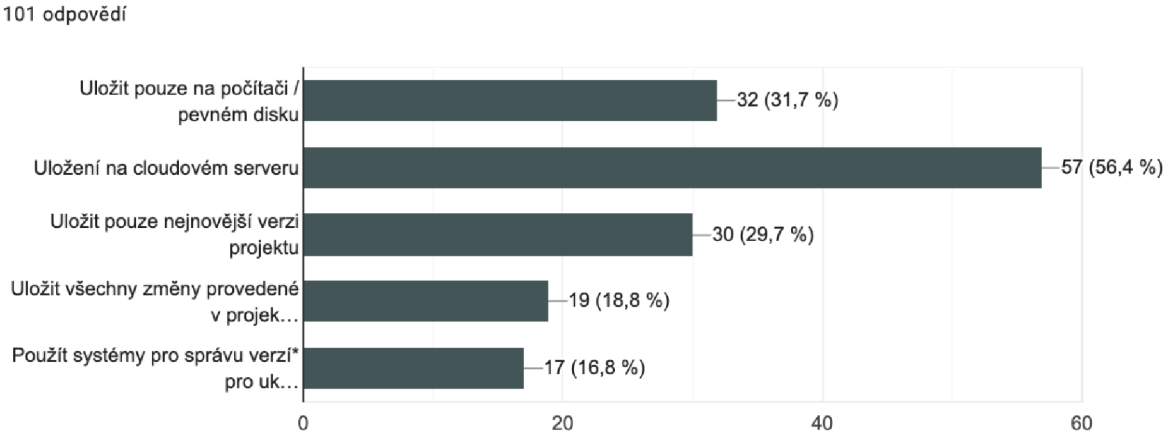
Obr. 41 Graf, přepracování projektů

Zdroj: Vlastní zpracování

Tato otázka měla ukázat, jak často nastala situace, kdy bylo nutné přepracovat část práce, a to s přihlédnutím k tomu, že poslední provedené změny byly chybné. V takové situaci je mnohem jednodušší vrátit se ke staré verzi dokumentu, než odstranit provedené změny a přepracovat je. To se stalo 58,4 % studentů a 41,6% v takové situaci nikdy nebylo.

Otázka č. 15: Představte si situaci, že se nemůžete osobně setkat s vaším partnerem, abyste společně pracovali na projektu (řekněme podmínky karantény). Jaký způsob uložení sdíleného souboru byste zvolili?

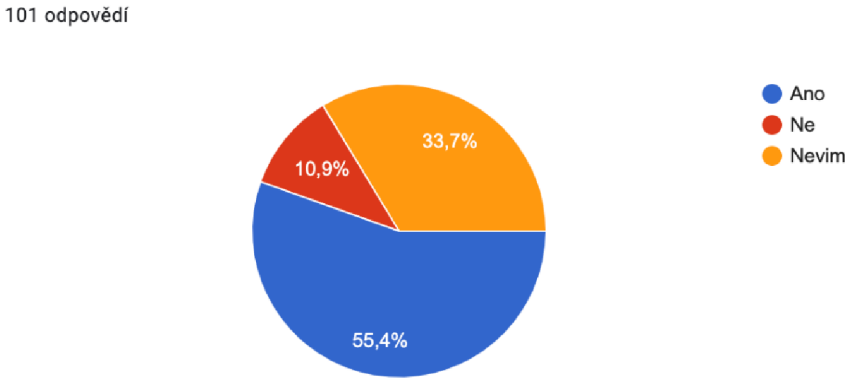
Na tuto otázku bylo možné zvolit několik odpovědí současně a zvolit nejen způsob ukládání dokumentů, ale také možnost uložit pouze jednu verzi nebo každou verzi s provedenými změnami. Tak 56,4 % lidí dalo přednost ukládání svých projektů na cloudový server, 31,7 % si vybralo úložiště na osobním počítači nebo pevném disku. Pokud jde o verze projektu, 29,7 % lidí si raději ponechá pouze jednu nejnovější verzi projektu a 18,8 % studentů si uloží všechny verze projektu se změnami.



Obr. 42 Graf, způsoby ukládání skupinových projektu

Zdroj: Vlastní zpracování

Otázka č. 16: Představte si, že máte možnost ukládat všechny své projekty zdarma s možností uložit každou jeho verzi (nejen osobní projekty, ale i skupinové, kde můžete sledovat změny provedené vašimi partnery). Dali byste přednost této metodě než obvyklému ukládání do počítače nebo na flash disk?



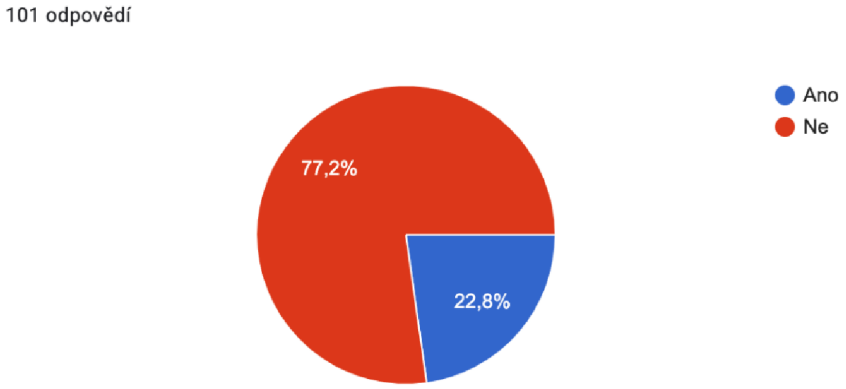
Obr. 43 Graf, využití VCS

Zdroj: Vlastní zpracování

Tato otázka byla nacílena, aby dozvědět, pokud by měl student příležitost použít technologii VCS, měl by o to zájem. A jak ukazuje graf, 55,4 % studentů by souhlasilo s jeho použitím, 33,7 % neví, zda by jej využili nebo ne, a pouze 10,9 % studentů by jej nechtělo používat.

Otázka č. 17: Setkali jste se s konceptem Systémy pro správu verzí (Version Control System, VCS)?

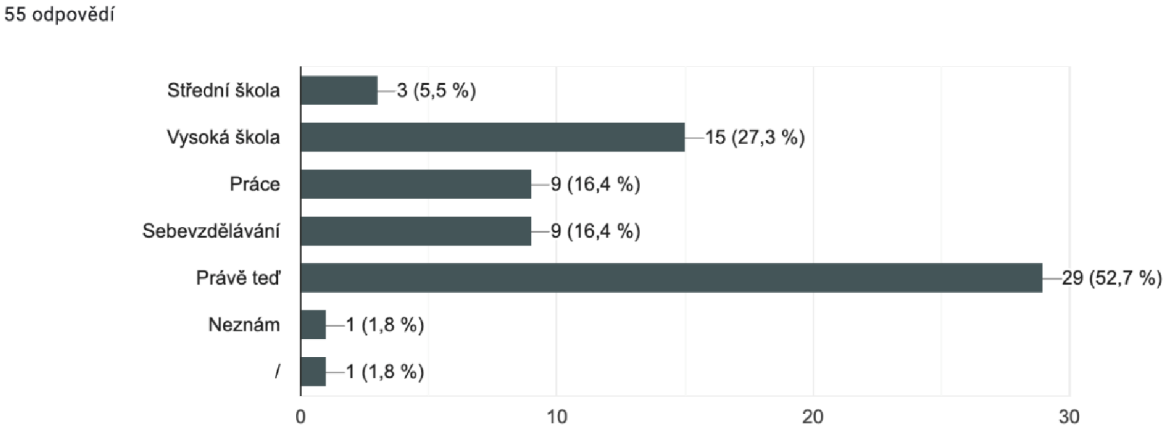
Tato otázka musela ukázat, jestli studenti znají pojem systémy pro správu verzí, a také bylo k otázce přidáno krátké vysvětlení tohoto pojmu. Bohužel pouze 22,8 % studentů tento termín znali a 77,2 % o něm nikdy neslyšelo.



Obr. 44 Graf, koncept VCS

Zdroj: Vlastní zpracování

<u>Otázka č. 18: Pokud ano, kde jste se o této technologii dozvěděli?</u>

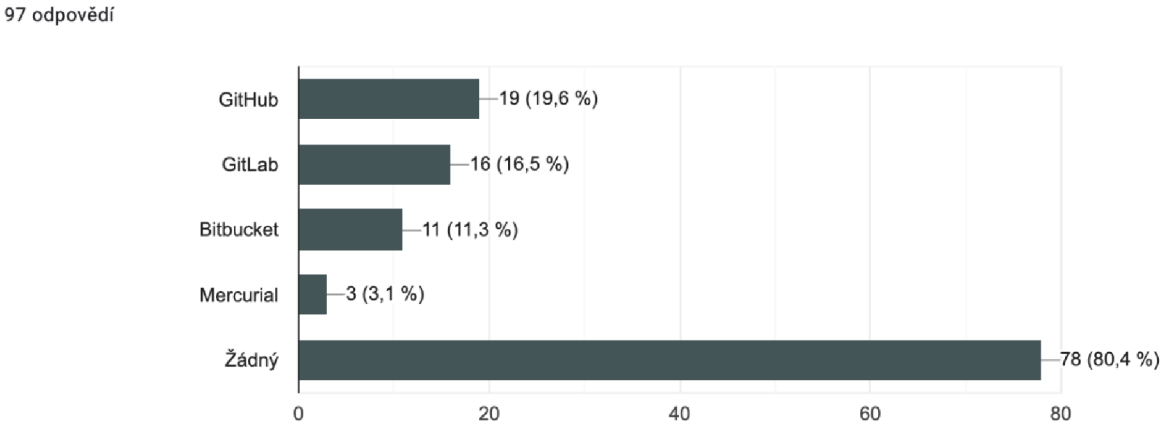


Obr. 45 Graf, ukazující, jak se studenti dozvěděli o VCS

Zdroj: Vlastní zpracování

Tato otázka byla volitelná, protože se předpokládalo, že ne všichni dotazovaní budou tento termín znát. Na otázku tedy odpovědělo pouze 55 lidí, z nichž 52,7 % studentů uvedlo, že se o této technologii dozvěděli až nyní (absolvováním tohoto testu), 27,3 % se o ní dozvědělo na vysoké škole, 16,4 % se o ní dozvědělo na pracovišti nebo prostřednictvím sebevzdělávání a 5,5 % na střední škole.

Otázka č. 19: Znáte níže uvedené servery?



Obr. 46 Graf, servery VCS technologie

Zdroj: Vlastní zpracování

Poslední otázka byla opět volitelná a shromáždila 97 odpovědí. Hlavním cílem bylo zjistit, který konkrétní server je v současné době mezi uživateli populárnější a známější. 80,4 % respondentů nezná žádný z uvedených serverů, 19,6 % zná GitHub, 16,5 % GitLub, 11,3 % BitBucket a 3,1 % Mercurial.

6.2 Využití GitHub Classroom na FIM UHK

Než se dotkneme praktického využití GitHub Classroom, ráda bych se podělila o své osobní zkušenosti, které se mi během studia na Univerzitě Hradec Králové podařilo získat. Protože je velmi důležité mít představu nejen ze strany učitelů, ale také ze strany studentů. A vzhledem k tomu, že v současné době se naše univerzita aktivně pohybuje v seznamech populárních vyšších institucí v zemi, je velmi důležité zůstat v trendech a podporovat propagaci inovativních technologií v oblasti vzdělávání. Stává se tak dobrým příkladem pro další mladé univerzity, ale také vyzývá univerzity, které učí studenty po celé generace. Koneckonců právě díky těmto výzvám dochází k největším objevům jak v oblasti vědy, tak v podnikání.

Strávila jsem na této univerzitě 5 let a díky vzdělávacímu systému, který poskytovala Česká Republika, jsem měla možnost vybrat si různé učitele, a tak jsem provedla svůj osobní výzkum toho, jak si univerzita realizuje sebe sama a moderní technologie. Podařilo se mi pozorovat uplatnění jak velmi starých praktik, tak touhu pedagogů přiblížit studentům zcela nové technologie v oblasti vzdělávání.

Univerzita tedy neustále hraje na kontrasty, protože učitel má možnost zvolit si metodu, kterou chce učit, a to na základě svých předchozích zkušeností a touhy. Tato metoda v tuto chvíli vypadá velmi dobře, protože je stejně snadné pro studenty přizpůsobit se. I přesto, že střední vzdělávací instituce stále striktně dodržují starý vzdělávací systém.

Na druhou stranu FIM a PřF patří mezi nejpokročilejší budovy a jejich technologický a pedagogický proces lze považovat za jeden z nejmodernějších. Protože při studiu exaktních věd je velmi důležité si správně vybavit a naplánovat proces učení. Dokonce i během pandemie byla organizace učitelů na vysoké úrovni. Ale ne každému připadalo snadné tyto změny přijmout. Když přecházíme na kompletní distanční vzdělávání, kde nám nejsou k dispozici technologické místnosti a laboratoře, je důležité nastavit k nim vzdálený přístup a zajistit bezpečnost při jejich používání.

Proto je naše univerzita opravdu na úrovni, protože absolventům je poskytován nejen vzdálený přístup, seznam programů, za které univerzita platí, a možnost ukládání našich souborů. Ale také využívá interaktivní výukové aplikace, což usnadňuje organizační proces jak pro studenty, tak pro učitele. Univerzita ale nikdy nezkoušela technologii, jako je Git-hosting.

A i když se mi během výuky podařilo setkat s učitelem, který projevil osobní iniciativu a navzdory programu kurzu dal několik praktických lekcí, aby řekl, jak používat GitHub, nebyla taková praxe nijak zvlášť účinná, protože kurz nebyl zaměřen na plné využití této technologie. A samotný proces byl velmi povrchní.

Avšak vzhledem k tomu, že hlavním směrem FIM je programování a práce s různými programovacími jazyky, nejméně 50 % studentů půjde pracovat v oboru, kde je znalost práce s platformami podobnými GitHubu obrovským privilegiem. Podle mého uvážení bude zahájení práce s GitHub Classroom dobrým rozhodnutím, protože v současné době se dokonale přizpůsobili práci se studenty a učiteli. A jsou připraveni pomáhat a propagovat jejich technologie na univerzitách. Na druhou stranu tato technologie není v České republice nijak zvlášť popularizována a v celé České republice jsou partnery programu GitHub Campus pouze 4 vzdělávací instituce.

Jak přesně lze takový nástroj na Univerzitě Hradec Králové použít?

- **Další způsob skladování školních materiálů.**

Každým rokem se zvyšuje počet studentů, kteří mají zájem o studium na naší fakultě, takže naše univerzita si musí vyřešit problém se zvýšením paměti na pevném disku. K dnešnímu dni má každý student 20 GB na uložení všech potřebných univerzitních materiálů. Hlavním bodem však je, že nákup disků SSD nebo HDD je velmi nákladná záležitost. V tomto případě máme další alternativu – cloudové úložiště.

Pokud se podíváme na dotazník, získáme následující informace, že studenti v průměru potřebují více než 1 GB paměti pro uložení školních projektů a také ne všechny vzdělávací instituce poskytují alespoň nějakou příležitost k ukládání materiálů na školních zdrojích. Nejprve není ani nutné být partnerem GitHub Classroom, aby každý student a učitel mohl na cloudovém serveru GitHub ukládat všechny možné materiály až do 2 GB. V případě partnerství, GitHub poskytují univerzitě slevu 25 % na jejich tarify, a tak můžeme zvýšit úložiště na 50 GB.

A takové množství paměti je velmi opodstatněné, protože máme možnost nejen uložit jeden soubor, ale celou jeho historii. Koneckonců, jak vyplývá z dotazníku, 58 % studentů muselo přepracovávat projekty s uvážením, že nové změny nejsou správné. V takovém případě je přepracování projektu poměrně zdlouhavý proces. GitHub nám na druhou stranu umožňuje uložit všechny změny a vrátit se ke starým verzím souboru, ale pouze pokud jsme je vložili do repozitáře.

- **Řízení kurzu**

V současné době je organizace naší univerzity na opravdu vysoké úrovni. Vzhledem k tomu, že kurz probíhá pomocí interaktivní výukové aplikace Blackboard Learn, která umožňuje nejen poskytovat materiály a informace studentům, ale také umožňuje přijímat projekty a provádět testy s předběžným hodnocením. Tato platforma však není ideální volbou pro programování, protože během praktických cvičení je spousta problémů s nastavením atd.

Tímto způsobem koukneme na takové kurzy, jako jsou například Algoritmy a datové struktury, Programování 1-2, Logické programování 1 atd. Protože jsou přímo spojeny s programováním a mohou sloužit jako skvělý začátek pro studium GitHub. Také v době studia těchto předmětů někteří učitelé prováděli průběh výuky a testování na papíře. Což podle mého uvážení není nejúčinnější možností.

Nejvýraznějším příkladem je zkouška Programování 2, kde musíte napsat program se specifickými funkcemi. Museli jsme poslat kód do učitelova e-mailu a po několika hodinách kontroly nás instruktor zavolal, abychom si promluvili o chybách atd. Nezní to tak špatně, ale ve srovnání s možným využitím GitHub bude celý proces probíhat mnohem rychleji. Samozřejmě budeme muset předem připravit celý úkol a automatické testy, ale ve výsledku bude celá zkouška mnohem rychlejší a pohodlnější. Můžeme tedy vytvořit soukromou třídu, ve které se úkoly budou otevírat, pouze když to učitel chce, student se připojí, stáhne úkol a okamžitě s ním začne pracovat. Po ukončení studia student odešle požadavek na vyžádání, po kterém okamžitě proběhne automatické vyhodnocení. Výsledky automatického testování samozřejmě nemohou být 100 % zárukou nesprávného rozhodnutí, ale výrazně ušetří čas. A tento princip práce je ideální pro jakýkoli předmět související s programováním.

- **Platforma pro spolupráci studentů**

Na základě dotazníku musí mnoho studentů neustále hledat možnost pohodlné spolupráce. Pokud se před pár lety mohli studenti setkat na univerzitě a pracovat společně na projektu, v současné situaci je taková možnost téměř nemožná. Studenti se snaží udržovat komunikaci a spolupráci prostřednictvím sociálních síti v kombinaci s úložnými servery. Jen velmi málo studentů používá existující platformy, které kombinují všechny tyto funkce. Jedním z důvodů, proč se nepoužívá, jsou neznalosti o jeho existenci a o tom, jak jej správně používat. Když se vrátíme k dotazníku, 77 % studentů o takové technologii nikdy neslyšelo. Největším probléem je nedostatečná popularita používání GitHubu ve vzdělávacích institucích.

Protože zaprvé je nutné vyškolit všechny učitele, jak tuto platformu používat, a za tímto účelem je nutné organizovat nějaké kurzy a podporu ze strany univerzity. Na druhé straně je nutné přepracovat výukový systém v konkrétních předmětech. Pokud předtím taková organizace vyžadovala spoustu zdrojů, pak s kurzem z GitHub Classroom, berou veškerou organizaci a podporu do svých rukou, organizováním školení nejen pro učitele, ale také pro studenty.

- **Příležitosti pro rozvoj velkých projektů**

V poslední době naše univerzita aktivně vyvíjí skvělé projekty a jedním z nich je FIM Bot. Učitelé aktivně podporují iniciativu studentů pracovat na projektech,

které budou mít praktické uplatnění, například jako 3D model univerzity, po němž se můžete pohybovat. A všichni víme, že projekty tohoto rozsahu vyžadují nejen týmovou práci, ale také nové nápady, neustálé hledání a řešení problémů a někdy nový pohled zvenčí.

S GitHub Classroom může být tento nápad mnohem snazší implementovat. Protože pomocí repozitáře může učitel snadno ovládat vývojový proces a také pomáhat v obtížných chvílích. Zveřejněním repozitáře mohou studenti také sdílet odkaz na úložiště, takže zájemci mohou použít náš kód a upozornit na slabá místa nebo naopak pomoci s nápadem. Prostřednictvím „Issues“ mohou také sledovat naše aktuální cíle a problémy a stát se tak součástí našeho projektu.

7 Seznam použité literatury

[1] About continuous integration. In: GitHub Docs [online]. San Francisco(CA): GitHub, Inc., c2008-2021. [cit. 2021-3-25]. Dostupné z: https://docs.github.com/en/actions/guides/about-continuous-integration

[2] About GitHub Education for students. GitHub Docs [online]. [cit. 2021-4-16]. Dostupné z: https://docs.github.com/en/education/explore-the-benefits-of-teaching-and-learning-with-github-education/about-github-education-for-students

[3] ARELIA, Jones. Set up your digital classroom with GitHub Classroom. The GitHub Blog [online]. 18. 3. 2020 [cit. 2021-4-6]. Dostupné z: https://github.blog/2020-03-18-set-up-your-digital-classroom-with-github-classroom/

[4] ASSAR, Vijith. The Software That Builds Software. The New Yorker [online]. 7. 8. 2013 [cit. 2020-6-23]. Dostupné z: https://www.newyorker.com/tech/annals-of-technology/the-software-that-builds-software

[5] BUHTIYAROV, Nikita. Краткий обзор GitHub и начало работы с ним. CodeX [online]. [cit. 2020-7-3]. Dostupné z: https://codex.so/github-start

[6] CHACON, Scott a Ben STRAUB. Pro git: Everything you need to know about Git. Second edition. Apress, 2014. ISBN 978-1484200773.

[7] Connect a learning management system to GitHub Classroom. GitHub Docs: Education [online]. [cit. 2021-4-11]. Dostupné z: https://docs.github.com/en/education/manage-coursework-with-github-classroom/connect-a-learning-management-system-to-github-classroom

[8] COPES, Flavio. A developer's introduction to GitHub. FreeCodeCamp [online]. [cit. 2020-6-23]. Dostupné z: https://www.freecodecamp.org/news/a-developers-introduction-to-github-1034fa55c0db/

[9] Create a group assignment. GitHub Docs: Education [online]. [cit. 2021-4-30]. Dostupné z: https://docs.github.com/en/education/manage-coursework-with-github-classroom/create-a-group-assignment

[10] Create an individual assignment. GitHub Docs: Education [online]. [cit. 2021-4-30]. Dostupné z: https://docs.github.com/en/education/manage-coursework-with-github-classroom/create-an-individual-assignment

[11] DOTSON, Kyt. GitHub Desktop arrives to replace Mac and Windows apps with unified GUI. SiliconANGLE [online]. 12. 8. 2015 [cit. 2021-3-25]. Dostupné z: https://siliconangle.com/2015/08/12/github-desktop-arrives-to-replace-mac-and-windows-apps-with-unified-gui/

[12] FELICIANO, Joseph, Margaret-Anne STOREY a Alexey ZAGALSKY. Student experiences using GitHub in software engineering courses. In: Proceedings of the 38th International Conference on Software Engineering Companion [online]. New York, NY, USA: ACM, 2016, 14. 5. 2016, s. 422-431 [cit. 2020-6-23]. ISBN 9781450342056. Dostupné z: doi:10.1145/2889160.2889195

[13] FIKSEL, Jacob, Leah R. JAGER, Johanna S. HARDIN a Margaret A. TAUB. Using GitHub Classroom To Teach Statistics. Journal of Statistics Education [online]. 2019, 27(2), 110-119 [cit. 2021-3-25]. ISSN 1069-1898. Dostupné z: doi:10.1080/10691898.2019.1617089

[14] GELBART, M. Teaching with GitHub. In: UBC MDS [online]. c2018-2021, 24. 08. 2017. Dostupné z: https://ubc-mds.github.io/2017-08-24-teaching-with-github/

[15] GENNARELLI, V. How to use pull requests in the classroom [online]. In: The GitHub Blog[online]. 29. 05. 2018. [cit. 2021-4-16]. Dostupné z: https://github.blog/2018-05-29-pull-requests-in-the-classroom/

[16] GitHub Classroom: README.md. GitHub [online]. [cit. 2021-3-18]. Dostupné z: https://github.com/education/classroom

[17] GitHub. Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2020-6-23]. Dostupné z: https://cs.wikipedia.org/wiki/GitHub

[18] GitLab против GitHub. Senior.ua: Только самое интересное [online]. г. Киев, c2018-2021. [cit. 2020-6-23]. Dostupné z: https://senior.ua/articles/gitlab-protiv-github

[19] GLASSEY, Richard. Adopting Git/Github within Teaching. In: *Proceedings of the ACM Conference on Global Computing Education* [online]. New York, NY, USA:

ACM, 2019, 9. 5. 2019, s. 143-149 [cit. 2020-6-23]. ISBN 9781450362597. Dostupné z: doi:10.1145/3300115.3309518

[20] HARDIN, J. GitHub Classroom. In: Teach Data Science [online]. c2018-2021, 13. 06. 2019. [cit. 2021-3-25]. Dostupné z: https://teachdatascience.com/gitclass/

[21] LARSÉN, Simon a Richard GLASSEY. RepoBee. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education [online]. New York, NY, USA: ACM, 2019, 2. 7. 2019, s. 534-540 [cit. 2021-3-25]. ISBN 9781450368957. Dostupné z: doi:10.1145/3304221.3319784

[22] Manage classrooms. GitHub Docs: Education [online]. [cit. 2021-4-11]. Dostupné z: https://docs.github.com/en/education/manage-coursework-with-github-classroom/manage-classrooms

[23] Mastering Issues. GitHub Guides [online]. 24. 7. 2020 [cit. 2020-6-23]. Dostupné z: https://guides.github.com/features/issues/

[24] MOSKALEVA, Y.P. a Z.S. SEIDAMETOVA. Teaching teamwork uses system control versions. Otkrytoe i distantsionnoe obrazovanie. 2018, (1(69), 12-17. ISSN 16095944. Dostupné z: doi:10.17223/16095944/69/2

[25] NGUYEN, S. Scholarly Git Experiences Part II: Git for Education. In: IASGE [online]. c2019-2021, 30. 09. 2019. [cit. 2021-3-7]. Dostupné z: https://investigating-archiving-git.gitlab.io/updates/git-for-education/

[26] OLSON, J. How to Teach Git Commits & GitHub to Teenagers. In: Upperline Code [online]. c2019-2021, 20. 03. 2020. [cit. 2020-6-23]. Dostupné z: https://blog.upperlinecode.com/how-to-teach-git-commits-github-to-teenagers/

[27] ORSINI, Lauren. Seven Ways To Use GitHub That Aren't Coding [online]. In: ReadWrite. c2003-2021, 8. 11. 2013. [cit. 2021-4-16]. Dostupné z: https://readwrite.com/2013/11/08/seven-ways-to-use-github-that-arent-coding/

[28] Quickstart for GitHub Educators. GitHub Docs: Education [online]. [cit. 2021-4-11]. Dostupné z: https://docs.github.com/en/education/quickstart

[29] REID, Karen L. a Gregory V. WILSON. Learning by doing. In: Proceedings of the 36th SIGCSE technical symposium on Computer science education - SIGCSE '05 [online]. New York, New York, USA: ACM Press, 2005, 2005, s. 272- [cit. 2020-6-23]. ISBN 1581139977. Dostupné z: doi:10.1145/1047344.1047441

[30] SWICEGOOD, T. Pragmatic Version Control Using Git. 1st edition. Pragmatic Bookshelf 2009.01.20. ISBN: 978-1934356159. Dostupné také z: http://index-of.es/Programming/Pragmatic%20Programmers/Pragmatic%20Version%20Control%20Using%20Git.pdf

[31] TANNER, G. An Introduction to Github Actions. In: Gabriel Tanner [online]. c2019-2021, 11. 03. 2019. [cit. 2021-3-25]. Dostupné z: https://gabrieltanner.org/blog/an-introduction-to-github-actions

[32] Types of GitHub accounts. GitHub Docs [online]. [cit. 2021-3-18]. Dostupné z: https://docs.github.com/en/github/getting-started-with-github/types-of-github-accounts

[33] Use autograding. GitHub Docs: Education [online]. [cit. 2021-4-30]. Dostupné z: https://docs.github.com/en/education/manage-coursework-with-github-classroom/use-autograding

[34] Why you should use GitHub: Lessons for the classroom and newsroom. In: Storybench [online]. c2011-2021, 29. 09. 2015. [cit. 2021-3-25]. Dostupné z: https://www.storybench.org/use-github-lessons-classroom-newsroom/

[35] WILEY, David. The Access Compromise and the 5th R – improving learning [online]. In: Open Content. c1998-2021, 05. 03. 2014. [cit. 2021-4-16]. Dostupné z: https://opencontent.org/blog/archives/3221

[36] ZAGALSKY, Alexey, Joseph FELICIANO, Margaret-Anne STOREY, Yiyun ZHAO a Weiliang WANG. The Emergence of GitHub as a Collaborative Platform for Education. In: Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing [online]. New York, NY, USA: ACM, 2015, 28. 2. 2015, s. 1906-1917 [cit. 2020-6-23]. ISBN 9781450329224. Dostupné z: doi:10.1145/2675133.2675284

[37] ГЕОРГИЕВСКАЯ, Татьяна. Системы контроля версий — основа командной разработки. Школа программирования ProgTips [online]. Москва, 2021, 26. 9. 2018 [cit. 2020-5-26]. Dostupné z: https://progtips.ru/instrumenty-programmista/sistemy-kontrolya-versij-osnova-komandnoj-razrabotki.html

[38] ЗАВЬЛОВА, Елена. Github: Что это такое и как его использовать. GitJournal [online]. 5. 2. 2020 [cit. 2020-6-23]. Dostupné z: https://gitjournal.tech/github-chto-jeto-takoe-i-kak-ego-ispolzovat/

[39] ЗЕЛЕНАЯ, Валерия. Работа с Git через консоль. HTML Academy: Блог Академии [online]. Санкт-Петербург, c2013-2021, 18. 2. 2021 [cit. 2021-3-18]. Dostupné z: https://htmlacademy.ru/blog/boost/frontend/git-console

[40] КУЛЕШОВ, М.В., САДОВ, Д.А., ТАЙМАСОВ, С.С. Системы контроля версий. In: Allbest [online]. c2000-2021, 28. 02. 2015. [cit. 2020-5-26]. Dostupné z: https://revolution.allbest.ru/programming/00521312_0.html

[41] КУРСОВАЯ РАБОТА: Системы контроля версий. NsmuBase [online]. c2014-2021, 17. 4. 2016 [cit. 2020-5-27]. Dostupné z: https://nsmubase.ru/kursovaya-rabota-sistemy-kontrolya-versij/

[42] ПЕРМЯКОВА, Е. Системы контроля версий. Allbest [online]. 2008, 13. 12. 2012 [cit. 2020-5-27]. Dostupné z: https://revolution.allbest.ru/programming/00229219_0.html#text

[43] ПОИСОВ, Дмитрий. Введение в системы контроля версий. HiT: Все о Hi-Tech [online]. c2009-2021 [cit. 2020-5-26]. Dostupné z: http://all-ht.ru/inf/prog/p_0_0.html

[44] ПОИСОВ, Дмитрий. Обзор систем контроля версий. HiT: Все о Hi-Tech [online]. c2009-2021 [cit. 2020-5-27]. Dostupné z: http://all-ht.ru/inf/prog/p_0_1.html

[45] Системы контроля версий, применяемые в процессе разработки программных продуктов. Allbest [online]. 19. 5. 2014 [cit. 2020-5-27]. Dostupné z: https://revolution.allbest.ru/programming/00411856_0.html

[46] Системы контроля версий. Studwood.ru [online]. c2017-2021 [cit. 2020-5-27]. Dostupné z: https://studwood.ru/1763534/informatika/vvedenie

[47] Что такое горячие клавиши и для чего они нужны? Компьютер для чайников: Компьютерная грамотность для начинающих [online]. 2021 [cit. 2021-3-18]. Dostupné z: https://www.pc-school.ru/chto-takoe-goryachie-klavishi-i-dlya-chego-oni-nuzhny/

8 Přílohy

1. Dotazník k výzkumu

GitHub

Technologie GitHub/GitHub Classroom a její aplikace ve vzdělávacích institucích

Dobrý den,

ráda bych Vás požádala o vyplnění dotazníku, který slouží k mé bakalářské práci na téma "Nástroj GitHub Classroom a možnosti jeho využiti". Dotazník je anonymní a jeho vyplnění Vám zabere jen pár minut.

Děkuji za Váš čas.

Oleksandra Naboichenko,
studentka Fakulty informatiky a managementu, Univerzity Hradec Králové

*Povinné pole

1. Jaké je Vaše pohlaví? *

- ○ Muž
- ○ Žena

2. Jaký je Váš věk? *

- ◯ 15 - 18 let
- ◯ 19 - 26 let
- ◯ 27 -35 let
- ◯ Více

3. Jste: *

- ☐ Student střední školy
- ☐ Student vysoké školy
- ☐ Zaměstnanec
- ☐ Jiná...

4. Jaký obor studujete? *

Text stručné odpovědi

5. Kolik za jeden semestr můžete mít seminárních prací?(projekty v textových dokumentech, praktické projekty jakéhokoli typu) *

- ◯ Žádný
- ◯ 1 - 2
- ◯ 3 - 4
- ◯ Více

6. Jakým způsobem získáváte osobní úkoly od učitelů? *

- [ ] Skupinový chat
- [ ] Obecné zasílání e-mailů
- [ ] Přes interaktivní výukovu aplikace(Blackboard Learn)
- [ ] Osobně
- [ ] Jiná...

7. Jak jste získávate zpětnou vazbu od učitele? *

- ( ) E-mail
- ( ) Interaktivní výuková aplikace
- ( ) Osobně
- ( ) Jiná...

8. Jste spokojeni s kvalitou zpětné vazby (1- nejnižší hodnocení, 5- nejvyšší hodnocení)? *

	1	2	3	4	5
Rychlá kontrola ...	( )	( )	( )	( )	( )
Dostatek kome...	( )	( )	( )	( )	( )
Odůvodnění ho...	( )	( )	( )	( )	( )
Kvalita konzulta...	( )	( )	( )	( )	( )

9. Stalo se vám, že učitel vaši práci nehodnotil(zapomněl/nemohl najít), nebo ji ztratil? *

- Ano
- Ne

10. Stalo se vám, že soubor s projektem byl poškozen v průběhu práce, v důsledku čehož data nebyla uložena nebo zcela ztracena? *

- Ano
- Nikdy

11. Máte ve zvyku multi-ukládání (ukládání důležitých dokumentů různými způsoby = ukládání do počítače, cloudu a sociální sítě současně)?

- Ne, není potřeba
- Ano, ale pouze s důležitými dokumenty
- Vždycky

12. Poskytuje vaše škola možnost ukládat vaše materiály na školní zdroje (místo na školním pevném disku, cloud, školní server)?

- Ano
- Ne

13. Kolik paměti(přibližně) vy jste potřebujete k uložení školních materiálů? *

- Pod 1 GB
- Nad 1 GB
- Více

14. Potřebujete pracovat na skupinových projektech? *

- ◯ Ano
- ◯ Ne

15. V případě skupinového projektu jakým způsobem preferujete výměnu provedených změn v souboru? *

- ☐ Pouze osobní schůzky a ukládání na osobním počítači
- ☐ Prostřednictvím sociálních sítí
- ☐ E-mailem
- ☐ Společný cloud
- ☐ Speciální server pro spolupráci
- ☐ Jiná...

16. Museli jste někdy přepracovat projekt vzhledem k tomu, že poslední provedené změny byly chybné? *

- ◯ Ano
- ◯ Nikdy

17. Představme si situaci, že se nemůžete osobně setkat s vaším partnerem, abyste společně pracovali na projektu (řekněme podmínky karantény). Jaký způsob uložení sdíleného souboru byste zvolili? *

*správa verzí je systém, který zaznamenává změny souboru nebo sady souborů v čase tak, abyste se mohli později k určité verzi vrátit.

- ☐ Uložit pouze na počítači / pevném disku
- ☐ Uložení na cloudovém serveru
- ☐ Uložit pouze nejnovější verzi projektu
- ☐ Uložit všechny změny provedené v projektu samostatně
- ☐ Použít systémy pro správu verzí* pro ukládání a provádění změn

18. Představte si, že máte možnost ukládat všechny své projekty zdarma s možností uložit každou jeho verzi(nejen osobní projekty, ale i skupinové, kde můžete sledovat změny provedené vašimi partnery). Dali byste přednost této metodě než obvyklému ukládání do počítače nebo na flash disku? *

- ◯ Ano
- ◯ Ne
- ◯ Nevim

19. Setkali jste se s konceptem Systémy pro správu verzí(Version Control System, VCS)? *

*správa verzí je systém, který zaznamenává změny souboru nebo sady souborů v čase tak, abyste se mohli později k určité verzi vrátit.

- ◯ Ano
- ◯ Ne

20. Pokud ano, kde jste se o této technologii dozvěděli?

- ☐ Střední škola
- ☐ Vysoká škola
- ☐ Práce
- ☐ Sebevzdělávání
- ☐ Právě teď
- ☐ Jiná...

21. Znáte níže uvedené servery?

- ☐ GitHub
- ☐ GitLab
- ☐ Bitbucket
- ☐ Mercurial
- ☐ Žádný
- ☐ Jiná...

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2019/2020

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

Jméno a příjmení:	**Oleksandra Naboichenko**
Osobní číslo:	**I1600584**
Adresa:	Horní Libchava 126, Horní Libchava, 47111 Horní Libchava u České Lípy, Česká republika
Téma práce:	Nástroj Github Classroom a možnosti jeho využití
Téma práce anglicky:	Github Classroom and its applications
Vedoucí práce:	doc. Ing. Filip Malý, Ph.D. Katedra informatiky a kvantitativních metod

Zásady pro vypracování:

Cílem je popsat technologii Github Classroom a popsat možnosti využití na FIM UHK za účelem správy projektů Osnova: 1. Úvod 2. Technologie git, služba Github 3. Github Classroom 3.1 Základní funkce a popis 3.2 Webové rozhraní (web-interface) 3.3 Využití GitHub Classroom na FIM UHK 4. Závěr 5. Literatura, zdroje

Seznam doporučené literatury:

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: