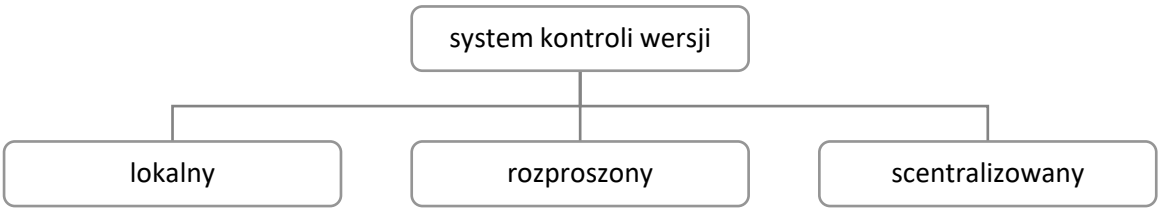
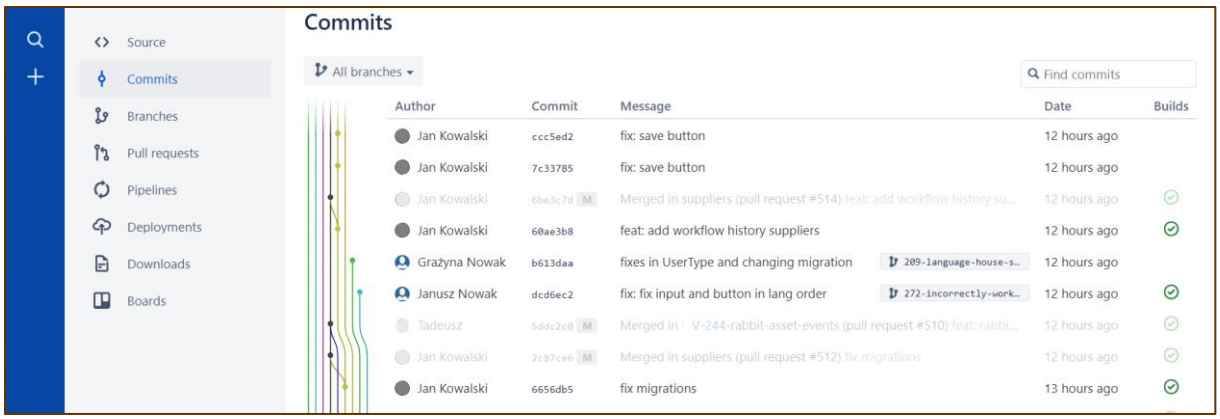
GIT - ROZPROSZONY SYSTEM KONTROLI WERSJI	PROJEKTOWANIE APLIKACJI INTERNETOWYCH

System kontroli wersji jest na dzień dzisiejszy standardowym narzędziem pracy każdego programisty. Jest to oprogramowanie służące do śledzenia zmian głównie w kodzie źródłowym oraz pomocy programistom w łączeniu zmian dokonanych w plikach przez wiele osób w różnym czasie. Tłumacząc tą definicję na prosty język system kontroli wersji tworzy historię zmian, których dokonaliśmy w kodzie. Nie ma w tym żadnej magii, wszystko opiera się na plikach tekstowych (w których jest Twój kod) i ich porównywaniu.

Istnieją trzy rodzaje systemów kontroli wersji:



Skupimy się na typie rozproszonym i najbardziej popularnym systemem kontroli wersji – Git (nie ważne, gdzie pójdziesz pracować na 98% będą używać tam Git'a). Rozproszony system kontroli wersji oznacza, że kod jest przytrzymywany lokalnie u każdego z deweloperów. Jeżeli wprowadzisz w kodzie jakieś zmiany robisz je lokalnie. Po zatwierdzeniu zmian wysyłasz kod do repozytorium, a następnie wszyscy pozostali programiści pobierają Twoje zmiany i mergują (łączą) je ze swoimi lokalnymi modyfikacjami.



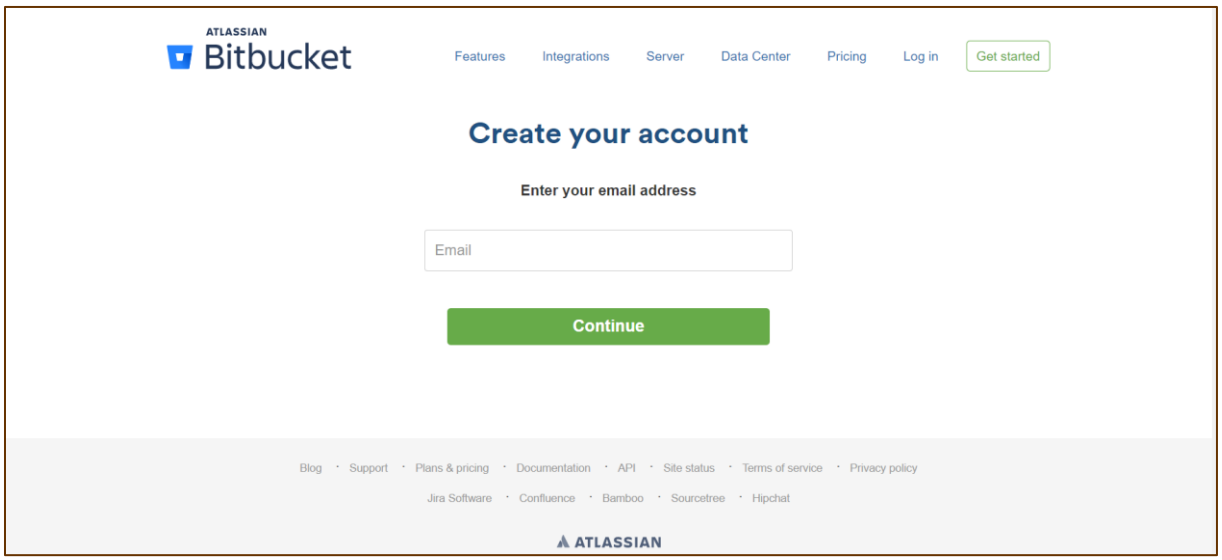
CO DAJE NAM HISTORIA ZMIAN W KODZIE?

Przede wszystkim możliwość wrócenia do poprzedniej wersji, gdy okaże się, że nowe zmiany coś popsuły lub celem nowych zmian było tylko sprawdzenie na szybko jakiegoś konceptu – jest to bardzo istotne i wykorzystuje się tą możliwość bardzo, bardzo często w codziennej pracy programisty. Możemy oczywiście również przeglądać co i kiedy robili nasi koledzy. Dodatkowo mamy dostęp do projektu z każdego miejsca, wystarczy jedynie pobrać kod repozytorium na dowolny komputer i uruchomić projekt.

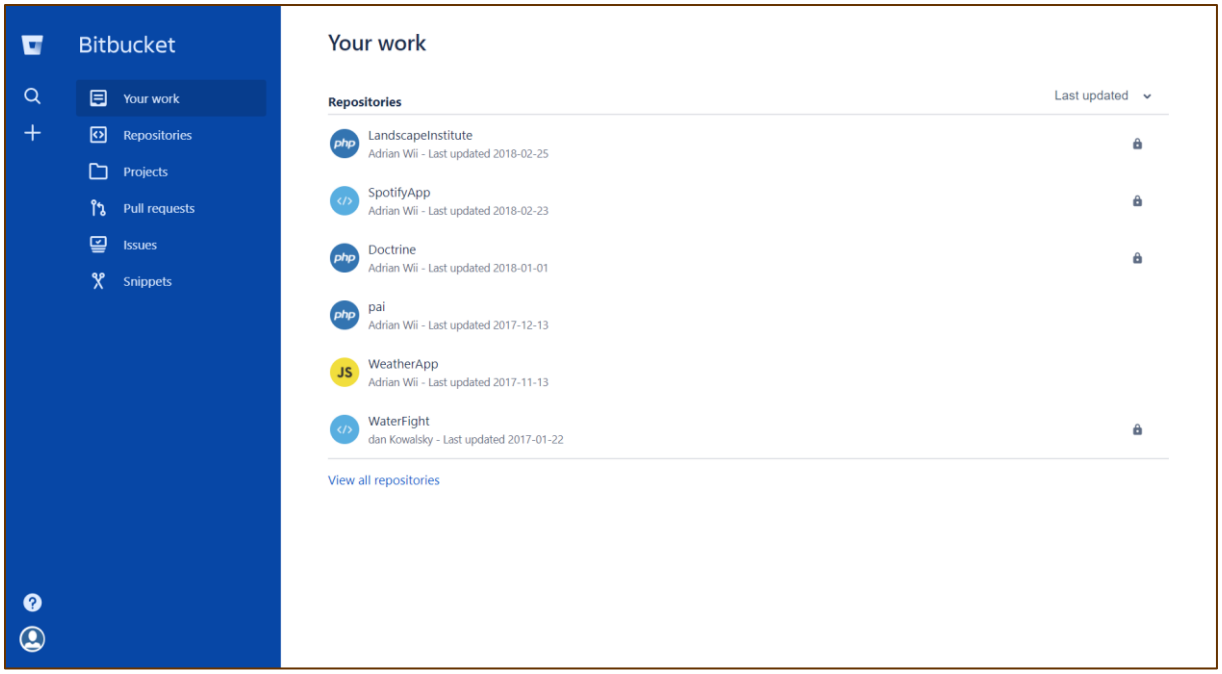
BITBUCKET – ZAŁOŻENIA KONTA

Bitbucket to rozwiązanie do obsługi repozytoriów Git dla profesjonalnych zespołów. Bitbucket jest bezpłatnym rozwiązaniem dla zespołów składających się z maksymalnie 5 osób. Pozwala przechowywać nieograniczoną liczba prywatnych repozytoriów o łącznej wielkości do 1GB na miesiąc. Narzędzie to pozwala również na podłączenie projektów z systemem zarządzania JIRA lub prostym w obsłudze, bezpłatnym trello.

W celu założenia konta wchodzimy na stronę https://bitbucket.org/account/signup/ Jeśli posiadasz konto na Bitbucket, wówczas możesz przejść do kroku zakładania repozytorium, który znajdziesz w dalszej części konspektu.

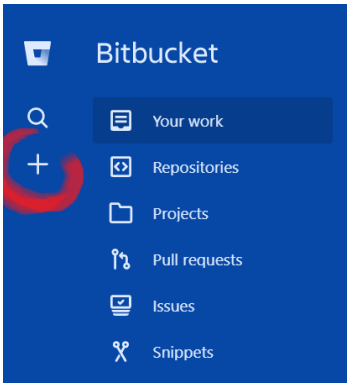


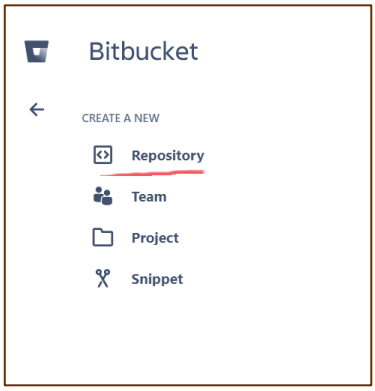
Przechodzimy poszczególne kroki. Po założeniu konta logujemy się, zostanie otwarta główna strona z stworzonymi projektami/ repozytoriami.



STWORZENIE REPOZYTORIUM

W kolejnym kroku naszym zadaniem będzie umieszczenie stworzonej wcześniej struktury projektu aplikacji internetowej w repozytorium. W tym celu tworzymy nowe repo. W tym celu klikamy + znajdujący się w lewej części ekranu. Następnie wybieramy **Repository.**



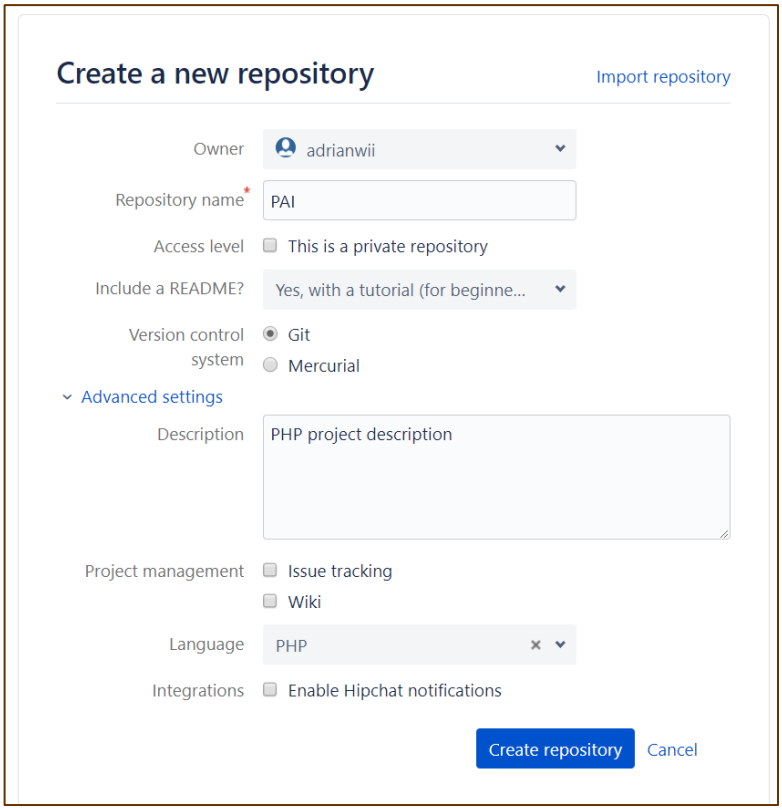


Następnie pojawi się okno z informacjami dotyczącymi tworzonego repozytorium. Możemy ustawić wiele opcji, skupy się jedynie na tych najważniejszych:

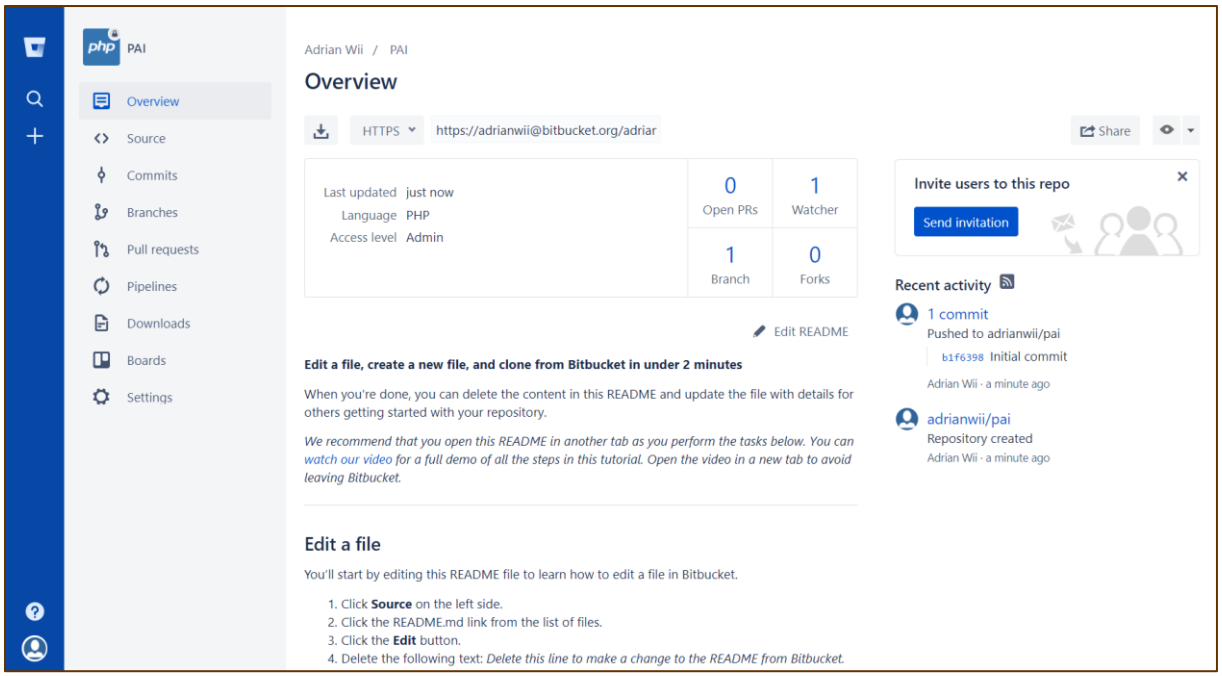
Repository name – tożsame z nawą projektu

Access level – na potrzeby przedmiotu stwórzmy repozytorium publiczne

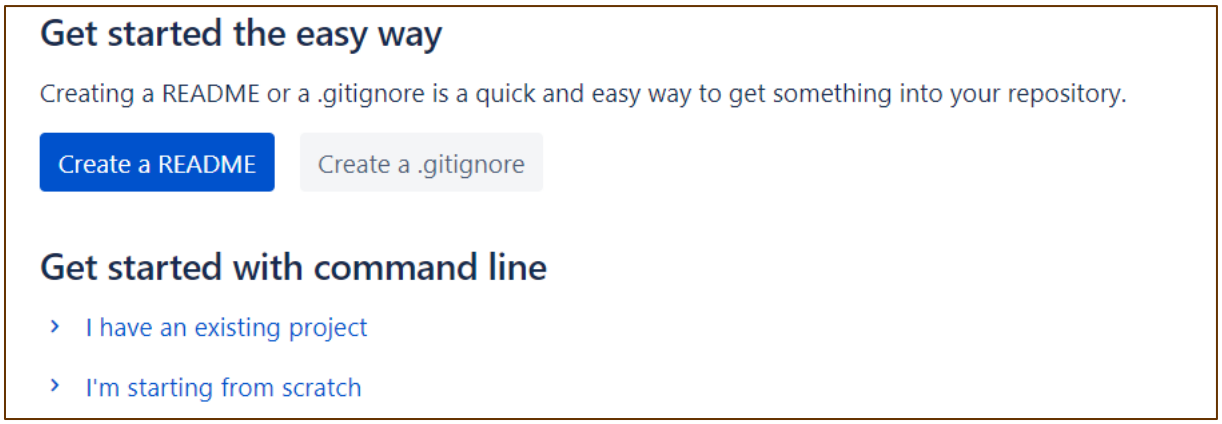
Advance settings – wpisujemy opis naszego projektu, oraz ustawiamy główny język. Możemy wybrać szereg technologii, ale możliwe jest zaznaczenie tylko jednej opcji - językiem głównym naszej aplikacji będzie PHP.



Po wybraniu przycisku **Create repository** zostaniemy przeniesieni na poniższa stronę:



Po stworzeniu repozytorium, możemy w łatwy sposób skonfigurować je używając linka z bitbacketu I **I have an existing project**



Lub skonfigurować całość z poniższymi krokami.

Naszym zadaniem będzie teraz zsynchronizowanie wcześniejszego szkieletu projektu z repozytorium. Przechodzimy do katalogu z projektem

cd Workspace/my_project

Aby utworzyć nowe repozytorium, użyjemy polecenia

git init

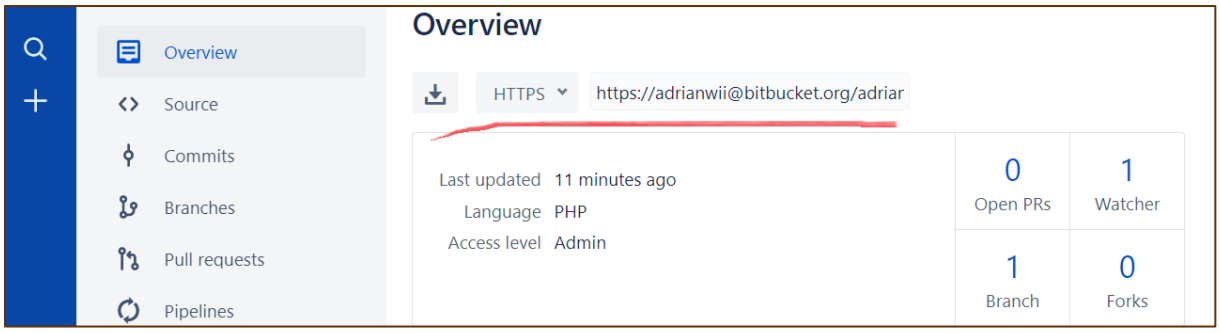
na ekranie powinieneś otrzymać podobny wynik

Initialized empty Git repository in C:/Users/project/pai/.git/

git init to jednorazowa komenda używana podczas początkowej konfiguracji nowego repo. Wykonanie tego polecenia spowoduje utworzenie nowego podkatalogu .git w bieżącym katalogu roboczym. Spowoduje to również utworzenie głównej gałęzi projektu master.

git remote add origin <link_do_repozytorium>

link do repozytorium podany jest bezpośrednio na stronie projektu przy HTTPS



dodajemy wszystkie pliki znajdujące się w projekcie, poprzez komendę

git add .

możemy sprawdzić, które pliki zostały dodane wywołując

git status

zostaną wyświetlone wszystkie pliki, których zmiany wcześniej nie były zatwierdzone (w naszym przypadku będą to wszystki pliki projektu). Teraz wystarczy wysłać jedynie nasze pliki na serwer.

Następnie dodajemy pierwszy **commit**. Dobrą konwencją jest pisanie wiadomości commitów po opcji **–m** z dużej liter – bez zakończenia kropką.

git commit -m "Initial commit"

pozostaje nam wysłać pierwsze zmiany na serwer

git push -u origin master

Oprócz konfigurowania zdalnego adresu URL repo, może być konieczne ustawienie globalnych opcji konfiguracji Git, takich jak nazwa użytkownika lub adres e-mail. Polecenie **git config** umożliwia skonfigurowanie instalacji Git (lub pojedynczego repozytorium) z poziomu wiersza poleceń.

git config --global user.name <name>

git config --local user.email <email>

W taki sposób dodaliśmy projekt do zdalnego repozytorium.

Bitbucket pozwala w łatwy sposób kontrolować pisany kod. Dodając osoby, można zapraszać ich do przeglądu kodu przed scaleniem zmian do gałęzi główne. Code review może bardzo poprawić jakość kodu. Jest tylko jeden warunek - by wykorzystać ten potencjał, trzeba je odpowiednio przeprowadzić, najlepiej przez osoby posiadające większe doświadczenie, które wyłapią programistyczne błędy i złe praktyki w naszym kodzie.

PRZEPŁYW PRACY Z UŻYCIEM GAŁĘZI

Gałęzie są używane do rozwijania funkcjonalności odizolowanych od siebie. Gałąź master jest domyślną gałęzią którą stworzyliśmy wraz z repozytorium.

Używaj innych gałęzi do rozwoju projektu, pozwala to na łatwiejsza kontrolę całości, podział projektu na podzadania, ale co najważniejsze pozwala to na prace z wieloma osobami. Możesz pracować na branchu o przykładowej nazwie security, zaś twój współpracownik rozwijający encję, będzie pracował na branchu entity. Gdy ktoś skończy pracę, w łatwy sposób złączy swój branch/ gałąź do głównego projektu.

Utwórz nową gałąź o nazwie "development" i przełącz się na nią

git checkout -b development

przełącz się z powrotem na master

git checkout master

i usuń gałąź

git branch -d development

gałąź nie jest dostępna dla innych dopóki nie wyślesz jej do zdalnego repozytorium

git push origin <branch>

Praktyka czyni mistrza! Stwórz nowy lokalny branch, na którym będziesz rozwijał projekt.

Gałąź master nie służy nigdy do rozwoju aplikacji, dlatego nie powinno na niej umieszczać się zmian. Jest to wersja z stabilnym, przetestowanym kodem, który gotowy jest do umieszczenia na serwerze.

AKTUALIZACJA I SCALANIE

Po dopisaniu części kodu, stworzeniu nowych plików, aby zaktualizować lokalne repozytorium do ostatniego commita, wykonaj

git pull

w swoim katalogu roboczym aby pobrać(fetch) i scalić(merge) zdalne zmiany.

aby scalić inną gałąź z gałęzią aktywną (np. master), użyj

git merge <branch>

w obu przypadkach git próbuje scalić zmiany automatycznie. Niestety nie zawsze jest to możliwe i powoduje konflikty. Jesteś odpowiedzialny za ich scalenie. Ręcznie poprzez edycję plików wskazanych przez git. Po zmianie musisz oznaczyć je jako scalone poprzez

git add <filename>

przed scaleniem zmian, możesz je obejrzeć używając

git diff <source_branch> <target_branch>

POBIERANIE ISTNIEJĄCEGO REPOZYTORIUM NA DYSK LOKALNY

Aby pobrać istniejące repozytorium musimy wykonać operację klonowania, która podobnie jak git init jest operacją jednorazową dla projektu. Gdy programista uzyska kopię roboczą, wszystkie operacje kontroli wersji są zarządzane za pośrednictwem lokalnego repozytorium.

git clone <repo url>

Standardowym zachowaniem jest ignorowanie plików, których w ogóle nie chcemy w swoim repozytorium. Są to pliki zawierające np. klucze, konfigurację bazy danych, konfigurację poczty email.

Do tego celu służą w git pliki tekstowe o nazwie .gitignore – plik ten zostaje utworzony z poleceniem git init.

Wystarczy dopisać odpowiedni wzorzecz, lub samą ścieżkę pliku, którego nie chcemy załączać, a system kontroli wersji nie będzie brał go pod uwagę przy wysyłaniu kolejnych zmian.