VERSIYA BOSHQARUV TIZIMLARI (GIT) VA JAMOA BILAN KODLASH STRATEGIYALARI

Maxammatyunusova Yulduzxon Dilmurot qizi[1], Narmanov Otabek Abdigapparovich[2], Yo'ldoshova Dilnoza Ilhomboy qizi[1], Madinabonu Mirxamidova Mirsaid qizi[1]

[1]TATU talabasi, [2]TATU dotsenti

E-mail: yunusovayulduz85@gmail.com; narmanov@tuit.uz; yoldoshovadilnoza00@gmail.com; madinabonumirxamidova14@gmail.com

Annotatsiya:

Ushbu maqolada zamonaviy dasturiy ta'minot ishlab chiqish jarayonida versiya boshqaruv tizimlarining, ayniqsa Git tizimining tutgan o‘rni yoritiladi. Git yordamida koddagi o‘zgarishlarni kuzatish, tarmoqlash (branching), birlashtirish (merging) kabi muhim funksiyalar samarali boshqariladi. Shuningdek, jamoaviy ishlashda kodni tartibli va xavfsiz saqlash, hamkorlikdagi ishni yengillashtirish uchun qo‘llaniladigan strategiyalar (masalan, Git Flow, trunk-based development) tahlil qilinadi. Maqola dasturchilar uchun amaliy tajriba va tavsiyalarni ham o‘z ichiga oladi.

Abstract:

This article explores the crucial role of version control systems in modern software development, with a particular focus on Git. It discusses how Git facilitates tracking changes in code, branching, merging, and collaboration among developers. Moreover, effective team coding strategies such as Git Flow and trunk-based development are analyzed. The article also provides practical tips and recommendations for developers working in collaborative environments.

Аннотация:

В данной статье рассматривается важная роль систем управления версиями в современном программировании, особенно системы Git. Описываются такие

ISSN 2181-3187

функции Git, как отслеживание изменений, ветвление, слияние и совместная работа в команде. Также анализируются эффективные стратегии командной разработки, включая Git Flow и trunk-based development. В статье приведены практические советы и рекомендации для разработчиков, работающих в коллективе.

Kalit so‘zlar:

Git, versiya boshqaruvi, jamoaviy kodlash, Git Flow, trunk-based development, pull request, branch, commit, merge.

Keywords:

Git, version control, team coding, Git Flow, trunk-based development, pull request, branch, commit, merge.

Ключевые слова:

Git, управление версиями, командная разработка, Git Flow, trunk-based development, pull request, ветка, коммит, слияние.

Introduction (Kirish). Zamonaviy dasturiy ta'minot ishlab chiqish jarayonlari murakkablik darajasining oshishi va jamoaviy ishlash zaruratining ortib borishi bilan ajralib turadi. Bunday sharoitda kod ustida bir nechta dasturchining bir vaqtning o‘zida ishlashi, o‘zgarishlarni kuzatib borish va muvofiqlashtirish muhim masalaga aylanadi. Shu nuqtayi nazardan, **versiya boshqaruv tizimlari (VCS)**, ayniqsa **Git**, dasturiy loyihalarni samarali yuritishning ajralmas qismiga aylandi.

Git — bu ochiq manbali, taqsimlangan versiya boshqaruv tizimi bo‘lib, u dasturchilarga koddagi har bir o‘zgarishni saqlash, tarixini kuzatish, turli filiallar (branch) ustida ishlash va jamoa bilan hamkorlikda muammosiz ishlash imkonini beradi. Ayniqsa, **Git Flow** kabi strategiyalar yordamida kodni strukturaviy boshqarish va ishlab chiqarish jarayonini tizimli tashkil etish mumkin.

ISSN 2181-3187

Ushbu maqolada Git tizimining asosiy imkoniyatlari, jamoaviy ishlashdagi afzalliklari hamda samarali kodlash strategiyalari yoritiladi. Shuningdek, real loyihalarda qanday tartibda ishlash kerakligi bo‘yicha tavsiyalar ham beriladi.

Git va uning asosiy imkoniyatlari

Git — Linus Torvalds tomonidan yaratilgan, kuchli va tezkor versiya boshqaruv tizimi hisoblanadi. U quyidagi asosiy funksiyalarni taklif etadi:

- **Commit** – kodga kiritilgan o‘zgarishlarni saqlash;
- **Branch** – yangi filial (yo‘nalish) yaratish orqali mustaqil kod ustida ishlash imkoniyati;
- **Merge** – turli branchlar ustida bajarilgan ishlanmalarni birlashtirish;
- **Revert/Reset** – noto‘g‘ri o‘zgarishlarni orqaga qaytarish;
- **Log** – loyihada qilingan barcha o‘zgarishlar tarixini ko‘rish.

Git taqsimlangan arxitekturaga ega bo‘lgani sababli, har bir dasturchi o‘zining lokal nusxasida ishlashi va keyinchalik o‘zgarishlarni asosiy repozitoriyga yuborishi mumkin. Bu esa jamoaviy ishlab chiqishda qulaylik yaratadi.

Jamoaviy kodlashdagi muammolar va ularning Git orqali yechimi

Jamoaviy ishlashda quyidagi muammolar yuzaga keladi:

- Kod ustida bir vaqtda ishlashdan kelib chiqadigan to‘qnashuvlar;
- Kutilmagan o‘zgarishlar va kod sifati pasayishi;
- Loyihaning versiyalarini boshqarishdagi chalkashliklar.

Git bu muammolarni hal qilish uchun quyidagi imkoniyatlarni beradi:

- **Pull request** (yoki merge request) orqali kodni asosiy tarmoqqa birlashtirishdan oldin ko‘rib chiqish va tasdiqlash;

ISSN 2181-3187

- **Branch naming conventions** — har bir vazifa uchun alohida branch ochish (masalan: feature/login-page, bugfix/header-error);
- **Conflict resolution** — to‘qnashuvlar paytida Git konfliktni aniqlab, uni hal qilish imkonini beradi.

Jamoaviy ishlash strategiyalari

A. Git Flow

Bu strategiyada quyidagi branchlar ishlatiladi:

- main — asosiy, barqaror kodlar saqlanadigan tarmoq;
- develop — barcha yangi funksiyalar birlashtiriladigan tarmoq;
- feature/* — yangi funksiyalar uchun tarmoqlar;
- release/* — yangi versiya chiqarishga tayyor holatdagi kodlar;
- hotfix/* — asosiy tarmoqdagi muhim xatolarni tezda tuzatish uchun.

Git Flow — o‘rta va yirik jamoalar uchun qulay strategiyadir.

B. Trunk-Based Development

Bu strategiyada barchaning kodlari bitta asosiy tarmoq (odatda main) bilan sinxronlashtiriladi va kichik commitlar orqali kod tez-tez yangilanib turadi. Bu usul Continuous Integration (CI) tizimlari bilan uyg‘un ishlaydi va kodni doimiy ishlab chiqarish holatida saqlaydi.

Amaliy tavsiyalar

- Har bir commit mazmunli va qisqa bo‘lishi kerak (masalan: fix: login button style).
- Branchlar ustida ishlaganda har kuni asosiy develop yoki main bilan sinxronlash kerak.

ISSN 2181-3187

- Katta o‘zgarishlardan oldin va keyin kodni tekshirish uchun pull request ochish tavsiya etiladi.
- .gitignore fayli orqali keraksiz fayllarni repozitoriyga kiritmaslik muhim.

Xulosa

Zamonaviy dasturiy ta’minot ishlab chiqish jarayonida versiya boshqaruv tizimlari, xususan Git, dasturchilar uchun muhim vosita bo‘lib xizmat qilmoqda. Git nafaqat koddagi o‘zgarishlarni kuzatish, balki jamoaviy ishlashda tartib, nazorat va xavfsizlikni ta’minlaydi.Kod sifati, ishlab chiqish tezligi va jamoa o‘rtasidagi muvofiqlik darajasi to‘g‘ridan-to‘g‘ri versiya boshqaruv tizimidan foydalanish madaniyatiga bog‘liq. Shuning uchun Git'ning barcha asosiy imkoniyatlarini chuqur o‘rganish va uni amaliyotda to‘g‘ri qo‘llash, ayniqsa jamoaviy loyihalarda, muvaffaqiyat kalitidir.

Foydalanilgan adabiyotlar ro’yxati

- Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress. https://git-scm.com/book/en/v2
- Loeliger, J., & McCullough, M. (2012). *Version Control with Git: Powerful tools and techniques for collaborative software development* (2nd ed.). O'Reilly Media.
- Vincent Driessen. (2010). *A successful Git branching model.* https://nvie.com/posts/a-successful-git-branching-model/