

ສາທາລະນະ​ລັດ ປະຊາທິປະ​ໄຕ ປະຊາຊົນ​ລາວ

ສັນຕິພາບ ​ເອກະລາດ ປະຊາທິປະ​ໄຕ ​ເອກະ​ພາບ ວັດທະນາ​ຖາວອນ

---------((0))---------



​ ມະຫາວິທະຍາໄລແຫ່ງຊາດ

ຄະນະວິທະຍາສາດທໍາມະຊາດ

ພາກວິຊາວິທະຍາສາດຄອມພິວເຕີ

ບົດລາຍງານ ວິຊາ ວິທີການຄົ້ນຄວ້າ

ສາຂາ (ຕໍ່ເນື່ອງ) ວິທະຍາສາດຄອມພິວເຕີ

ຊື່​ບົດຈົບຊັ້ນ​ (Title)

ພາສາ​ລາວ: ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ອອນລາຍ

ພາສາ​ອັງກິດ: Southern Bus Ticket Online Booking System

ສະມາຊິກໃນກຸ່ມ (Project Team)

|  |  |  |  |
| --- | --- | --- | --- |
| ລ/ດ | ລະຫັດ | ຊື່ ແລະ ນາມສະກຸນ | ເບີໂທ |
| 1 | 204N0025.19 | ທ ພອນຄຳ ແກ້ວມະນີ | 020 28022677 |
| 2 | ​204N0002.19 | ທ ມະໂນພອນ ມະໂນກຸນ | 020 54000003 |

ສອນໂດຍ ປອ. ລັດສະໝີ ຈິດຕະວົງ

1. ຄວາມສຳຄັນຂອງບັນຫາ

ໃນຍຸກປັດຈຸບັນເຕັກໂນໂລຊີຂໍ້ມູນຂ່າວສານມີຄວາມກ້າວໜ້າ ແລະ ທັນສະໄໝຫຼາຍຂຶ້ນເປັນຕົ້ນແມ່ນເຄື່ຶຂ່າຍ Internet ໄດ້ຂະຫຍາຍຕົວຢ່າງວ່ອງໄວ ຈນກ້າວເຂົ້າສູ່ບົດບາດ ແລະ ຄວາມສຳຄັນຢ່າງຫຼວງຫຼາຍ. ເຮັດໃຫ້ມີການຄິດຄົ້ນສິ່ງອຳນວຍຄວາມສະດວກເພື່ອພັດທະນານຳໃຊ້ໃນຊີວິດປະຈຳວັນຂອງພວກເຮົາ ບໍ່ວ່າຈະເປັນດ້ານການຕິດຕໍ່ສື່ສານ, ການປະຊາສຳພັນຂ່າວ, ການໂຄສະນາ, ການສຶກສາ, ສື່ສານຕ່າງໆ ແລະ ຕະຫຼອດຮອດການຄ້າຂາຍເປັນຕົ້ນກໍ່ໄດ້ນຳເອົາເຕັກໂນໂລຊີເຂົ້າມາຊ່ວຍໃນການດຳເນີນງານໃຫ້ສະດວກສະບາຍວ່ອງໄວ, ນອກຈາກນັ້ນຍັງສາມາດເຜີຍແຜ່ຂໍ້ມູນຂ່າວສານຕ່າງໆຜ່ານເວັບໄຊຕ່າງໆ. ເຮັດໃຫ້ເວັບໄຊໃນເຄື່ອຂ່າຍໃນອິນເຕີເນັດສາມາດຕອບສະໜອງໃນດ້ານຕ່າງໆຜ່ານເວັບໄຊຕ່າງໆ. ເຮັດໃຫ້ເວັບໄຊໃນເຄື່ອຂ່າຍໃນອິນເຕີເນັດສາມາດຕອບສະໜອງໃນດ້ານຕ່າງໆທີ່ກ່າວມານັ້ນເປັນຢ່າງດີ ແລະ ຈະສັງເກດໄດ້ດ້ວຍໜ່ວຍງານອົງກອນທຸລະກິດ, ບໍລິສັດ ຫຼື ຮ້ານຄ້າໃຫ່ຍຕ່າງໆ ຈະມີຄວາມສົນໃຈ ແລະ ແນໃສ່ຄວາມສຳຄັນຂອງສິນຄ້າຫຼາຍຂຶ້ນ ຈຸດປະສົງເພື່ອປະຊາສຳພັນ ຫຼື ການຄ້າຂາຍ, ການສ້າງເວບໄຊຂອງບໍລິສັດ ເພື່ອບໍລິຫານການຂາຍສິນຄ້າກໍຖືວ່າເປັນທາງເລືອກໜຶ່ງທີ່ທັນສະໄໝ ເປັນການອຳນວຍຄວາມສະດວກໃຫ້ແກ່ພະນັກງານ ແລະ ເຈົ້າຂອງບໍລິສັດ, ເປັນການໂຄສະນາບໍລິສັດຜ່ານທາງເວບໄຊໃຫ້ເປັນທີ່ຮູ້ຈັກກັນຫຼາຍຂຶ້ນ ເປັນທາງເລືອກໜຶ່ງທີ່ສະດວກສະບາຍໃນການບໍລິ ຫານການຂາຍທີ່ລູກຄ້າສາມາດເລືອກຊື້ ແລະ ເບິ່ງສິນຄ້າພາຍໃນບໍລິສັດໄດ້.

ສະຖານີຂົນສົ່ງໂດຍສານສາຍໃຕ້ ເປັນບ່ອນບໍລິການຮັບ-ສົ່ງຜູ້ໂດຍສານ, ສິນຄ້າວັດຖຸສິ່ງ ຂອງ ແລະ ສັດ ຈາກຈຸດໜຶ່ງໄປຫາອີກຈຸດໜຶ່ງ ຊຶ່ງສະຖານີຂົນສົ່ງໂດຍສານທາງໄກສາຍໃຕ້ນີ້ແມ່ນໄດ້ ສ້າງຕັ້ງຂຶ້ນໃນວັນທີ 1 ກັນຍາ 2016 ເຊິ່ງມີລາຍລະອຽດດັ່ງລຸ່ມນີ້: ທີ່ຕັ້ງ ແລະ ພາ ລະບົດບາດຂອງສະຖານນີຂົນສົ່ງທາງໄກສາຍໃຕ້ ແມ່ນສະຖານີໜຶ່ງຊຶ່ງຕັ້ງຢູ່ບ້ານສະພັງມຶກ, ເມືອງ ໄຊທານີ, ນະຄອນຫຼວງວຽງຈັນ, ຖະໜົນເລກທີ 450 ປີ ໃກ້ກັບ ສີ່ແຍກໄຟແດງດົງໂດກ. ສະຖານນີຂົນສົ່ງທາງໄກສາຍໃຕ້ ປະກອບມີຫຼາຍໜ່ວຍງານຄື: ອຳນວຍການໃຫ່ຍມີ 1 ທ່ານ, ເລຂານຸການມີ 1 ທ່ານ, ໜ່ວຍງານແຜນການມີ 1 ທ່ານ, ໜ່ວຍງານຮັບ-ຈ່າຍເງິນມີ 1 ທ່ານ, ໜ່ວຍງານບໍລິການຂາຍປີ້ມີ 7 ທ່ານ, ໜ່ວຍງານຮັກສາຄວາມປອດໄພມີ 6 ທ່ານ,ໜ່ວຍງານບໍລິ ການເຮືອນພັກມີ 8 ທ່ານ ແລະ ບັນດາບໍລິສັດ ທິ່ເຂົ້າມາດໍາ ເນີນທຸລະກິດ ໃນສະຖານຂົນສົ່ງໂດຍສານປະກອບມີ: ບໍລິສັດ ຂົນສົ່ງໂດຍສານຈິດປະສົງ ຍອດນິຍົມ, ບໍລິສັດ ແສງສົມ ບູນ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ແສນສະບາຍ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ຈັນທະຈອນ ຂົນສົ່ງໂດຍ ສານ, ບໍລິສັດ ຈໍາປາສັກ ຂົນສົ່ງໂດຍສານ, ບໍລິສັດ ແສງຈະເລີນ ລົດຕຽງນອນ, ບໍລິສັດ ກຽງໄກ VIP, ບໍລິສັດ ສີທອນ ພວງປະເສີດ ລົດຕຽງນອນ. ນອກຈາກນີ້ ສະຖານຍີງ ມີສະຖານທີ່ພັກ , ຮ້ານຄ້າ, ຮ້ານຂາຍຍ່ອຍ, ຮ້ານອາຫານ ແລະ ສິ່ງອໍານວຍຄວາມສະດວກຕ່າງໆ ໄວ້ເພື່ອບໍລິການຜູ້ໂດຍສານທີ່ມາລໍຖ້າລົດໄປຈຸດໝາຍປາຍທາງ.

ເນື່ອງຈາກວ່າ ຈັດການຂໍ້ມູນ, ລາຍງານຂໍ້ມູນຕ່າງໆ, ລວມທັງການຂາຍປີ້ແມ່ນຍັງໃຊ້ແບບຈົດ ແລະ ຜູ້ໂດຍສານທີ່ຕ້ອງການຈອງປີ້ລວງໜ້າ ຕ້ອງໄດ້ໂທຫາພະນັກງານຂາຍປີ້ເພື່ອຈອງ ເຊິ່ງເຮັດໃຫ້ການບໍລິການມີການຊັກຊ້າ ແລະ ຂໍ້ມູນຍັງມີການຕົກເຮ່ຍເສຍຫາຍ

ດັ່ງນັ້ນ , ພວກຂ້າພະເຈົ້າຈຶ່ງເຫັນຄວາມສໍາຄັນຂອງບັນຫາ ຈຶ່ງມີແນວຄວາມຄິດທີ່ຈະສ້າງລະບົບຈອງປີລົດເມ ແບບອອນໄລ ຂອງສະຖານີຂົ່ນສົ່ງໂດຍສານສາຍໃຕນັ້ນຂຶ້ນມາ ເພື່ອຊ່ວຍຫຸດຜ່ອນຄວາມຫຍຸ້ງຍາກໃນການຈອງປີ້ລົດ, ຈັດເກັບຂໍ້ມູນ, ຫຸດຜ່ອນຄວາມຊັກຊ້າໃນການຈັດການຂໍ້ມູນ, ເຮັດໃຫຂໍ້ມູນມີຄວາມເປັນລະບຽບຮຽບຮ້ອຍ ແລະ ເພືອໃຫ້ມີຄວາມສະດວກວ່ອງໄວຕໍ່ການຄົົນຫາຂໍ້ມູນ.

1. ຈຸດປະສົງຂອງການຄົ້ນຄ້ວາ (Objectives)

* ເພື່ອສຶກສາບັນຫາທີ່ເກີດຂຶ້ນໃນປະຈຸບັນ ແລະ ຄວາມຕ້ອງການຂອງລະບົບ.
* ເພື່ອສ້າງລະບົບຂາຍປີ້ລົດອອນໄລຂອງສະຖານີຂົນສົ່ງໂດຍສານສາຍໃຕ້.
* ເພື່ອສ້າງຮູບແບບການຈັດການຂໍ້ມູນການໃຫບໍລິການ.
* ເພື່ອເຜີຍແຜ່ຂໍ້ມູນການຂາຍປີ້ລົດເມຂອງສະຖານຂົນສົ່ງໂດຍສານສາຍໃຕ້.
* ເພື່ອການລາຍງານໃຫ້ສະດວກ ແລະ ຖືກຕ້ອງ.

1. ຂອບເຂດການຄົ້ນຄ້ວາ (Scope)

ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ແບບອອນລາຍ ເປັນລະບົບແບບ Client-Server Web Application ເຊິ່ງປະກອບດ້ວຍໜ້າວຽກຫຼັກດັ່ງນີ້:

* ຈັດການຂໍ້ມູນພື້ນຖານ : (ຂໍ້ມູນພະນັກງານ, ຂໍ້ມູນລົດ, ຂໍ້ມູນປະເພດລົດ, ຂໍ້ມູນສາຍທາງ)
* ສະໝັກສະມາຊິກ
* ບໍລິການ : (ຈອງປີ້, ອອກປີ້)
* ລາຍງານ : (ລາຍງານຂໍ້ມູນການຈອງ, ລາຍງານຂໍ້ມູນພະນັກງານ, ລາຍງານຂໍ້ມູນສາຍທາງ, ລາຍງານຂໍ້ມູນລົດ, ລາຍງານຂໍ້ມູນອອກປີ້)

1. ປະໂຫຍດຄາດວ່າຈະໄດ້ (Expected Outcome of the Project)

* ໄດ້ລະບົບຈອງປີ໊ລົດເມແບບອອນລາຍຂອງສະຖານີຂົ່ນສົ່ງສາຍໃຕ້
* ໄດ້ລະບົບທີ່ຈະຊ່ວຍແກ້ໄຂບັນຫາການຈອງຈອງໄດ້ສະດວກ ແລະ ວ່ອງໄວຂຶ້ນ
* ໄດ້ລະບົບຊ່ວຍເພີ່ມຊ່ອງທາງໃນການຂາຍປີ້ໃຫ້ກັບຜູ້ປະກອບການ
* ມີລະບົບເຜີຍແຜ່
* ໄດ້ລະບົບທີ່ສາມາດສ້າງລາຍງານໄດ້ຢ່າງສະດວກ ແລະ ຖືກຕ້ອງ

1. ວິທີດຳເນີນການຄົ້ນຄ້ວາ (Research Methodology)

ຂັ້ນຕອນວິທີການດຳເນີນການພັດທະນາລະບົບໃນຄັ້ງນີ້ແມ່ນອີງໃສ່ລັກສະນະແບບຈຳລອງຂອງ Adapted Waterfall Model ເປັນແນວທາງໃນການວິເຄາະ ແລະ ອອກແບບລະບົບ ເຊິ່ງມີໜ້າວຽກດັ່ງນີ້:

ວາງແຜນ

ວິເຄາະ

ອອກແບບ

ພັດທະນາ

ທົດສອບ

ສ້າງເອກະສານ

ຮູບທີ່ 1 ວົງຈອນການພັດທະນາແບບນ້ຳຕົກ (Adapted Waterfall Model)

* + ໄລຍະການວາງແຜນ

ດຳເນີນການຈັດຕັ້ງກຸ່ມຂຽນບົດໂຄງການພຽງພ້ອມກັບການກຳນົດຂໍ້ຂອງໂຄງການກຳນົດຫົວຂໍ້ຂອງໂຄງການ. ຈາກນັ້ນ, ກໍໄດ້ລົງເກັບກຳຂໍ້ມູນຢູ່ສະຖານນີຂົຍສົ່ງໂດຍສານສາຍໃຕ້ ເພື່ອໃຫ້ໄດ້ຂໍ້ມູນ ແລະ ຂັ້ນຕອນການເຮັດວຽກໂດຍລວມກ່ຽວກັບການເຮັດວຽກຕ່າງໆຂອງສະຖານີດັ່ງກ່າວ.

* + ໄລຍະການວິເຄາະ

ໄລຍະນີ້ພວກເຮົາຈະສຶກສາລະບົບເກົ່າ ແລະ ຄວາມຕ້ອງການຂອງຜູ້ໃຊ້. ຈາກນັ້ນ, ກໍນຳເອົາຂໍ້ມູນຕ່າງໆລວບລວມເອົາຂໍ້ມູນຕ່າງໆທີ່ລວບລວມໄດ້ມາເພື່ອວິເຄາະເປັນຂໍ້ກຳນົດຄວາມຕ້ອງການຂອງລະບົບໃໝ່ພ້ອມທັງແຕ້ມແບບຈຳລອງ DFD ເພື່ອສະແດງເຖິງການໄຫຼຂໍ້ມູນໄປເຖິງຂະບວນການ ແລະ ແຜນວາດ E-R Diagram ເພື່ອສະແດງຄວາມສຳພັນລະຫວ່າງຂໍ້ມູນ.

* + ໄລຍະການອອກແບບ

ໄລຍະການອອກແບບ ແມ່ນໄລຍະທີ່ພວກຂ້າພະເຈົ້າ ນໍາເອົາຜົນທີ່ໄດ້ຈາກການວິເຄາະມາອອກແບບລະບົບໂດຍການອອກແບບ, ໜ້າຟອມປ້ອນຂໍ້ມູນ ແລະ ຫນ້າຟອມລາຍງານໂດຍນໍາໃຊ້ Visual Studio code ລວມທັງອອກແບບຖານຂໍ້ມູນ.

* ໄລຍະການພັດທະນາ

ດຳເນີນການສ້າງຂໍ້ມູນຕາມທີ່ໄດ້ເຮັດ Data Dictionary ໃນຂັ້ນຕອນກ່ອນໜ້ານັ້ນ. ຈາກນັ້ນ, ກໍລົງມືປະຕິບັດຂຽນ Code ເພື່ອຕິດຕໍ່ ແລະ ເຮັດວຽກກັບຖານຂໍ້ມູນທີ່ໄດ້ສ້າງໄວ້ແລ້ວ.

* + ໄລຍະການທົດສອບ

ສ້າງແບບວິທີການທົດສອບລະບົບ ແລະ ນຳເອົາຊຸດຂໍ້ມູນທີ່ນຳມາທົດສອບຢ່າງໜ້ອຍ 20 records ມາດຳເນີນການທົດສອບແລ້ວສະຫຼຸບຜົນໄດ້ຮັບ ແລະ ທຳການປັບປຸງແກ້ໄຂຈົນໄດ້ຜົນອອກທີ່ຖືກຕ້ອງ

* ໄລຍະການສ້າງເອກະສານ

ສ້າງປຶ້ມບົດຈົບຊັ້ນ ລະບົບການຈອງປີ້ແບບອອນລາຍຂອງສະຖານີຂົນສົ່ງໂດຍສານສາຍໃຕ້ ແລ້ວສົ່ງມອບປຶ້ມພ້ອມທັງແຜ່ນ CD ທີ່ປະກອບດ້ວຍ Source Code ,ເອກະສານບົດຈົບຊັ້ນ ແລະ Presentations.

1. ສະຖານທີ່ສຶກສາ (Study Site)

ສະຖານີຂົນສົ່ງ ໂດຍສານວຽງຈັນ ຈຳກັດ (ສາຍໃຕ້-ຕ່າງປະເທດ)

1.  ​ໄລຍະປະຕິບັດ (Duration)
2. ເຄື່ອງມື​​ທີ່ນຳໃຊ້​ໃນ​ການ​ພັດທະນາ (Development Tools)

|  |  |
| --- | --- |
| 1. Hardware | 1. Software |
| * ຄອມພິວເຕີ Lenovo Intel(R) Core(TM) I5-3230M CPU @2.60GHz, RAM 8GB DDR3L 1600MHz, SSD 240GB,HDD 500GB * USB 16GB 1ອັນ | * ລະບົບປະຕິບັດການ Windows 10 Professional 64 Bit * Microsoft Visio 2016 ໃຊ້ແຕ້ມແຜນວາດການໄຫຼຂໍ້ມູນ (DFD, ER, Flowchart) * Adobe XD ໃຊ້ອອກແບບ UX/UI * Studio 3T For MongoDB ແລະ Moon Modeler ໃຊ້ຈັດການຖານຂໍ້ມູນ ແລະ ອອກແບບ Database Model * Visual Studio Code ໃຊ້ຂຽນໂຄດດ້ວຍພາສາ JavaScript (ReactJS, NodeJS, GraphQL) * MS Office 2013 Professional ໃຊ້ເພື່ອສ້າງເອກະສານຕ່າງໆ, ບົດນຳສະເໜີ ແລະ ເຮັດປຶ້ມບົດຈົບຊັ້ນ |

1. ເອກະສານອ້າງອີງ (Reference)

ສົມມິດ ທຸມມາລີ, ແລະ ອາມອນ ຈັນທະພາວົງ. (2012). ການວິເຄາະ ແລະ ອອກແບບລະບົບ (System Analysis and Design), ໜ້າ 56, 68 ນະຄອນຫຼວງວຽງຈັນ: ໂຮງພິມນະຄອນຫຼວງວຽງຈັນ

1. ທົບທວນທິດສະດີ ແລະ ບົດທີ່ກ່ຽວຂ້ອງ
   1. ທົບທວນທິດສະດີທີ່ກ່ຽວຂ້ອງ

ລະບົບຈອງປີ້ລົດເມສາຍໃຕ້ອອນລາຍແມ່ນພັດທະນາຂຶ້ນໂດຍນຳໃຊ້ທິດສະດີ ຫຼື ຄວາມຮູ້ຈາກ 3 ສ່ວນຄື: ທິດສະດີໃນການວິເຄາະ ແລະ ອອກແບບລະບົບ, ທິດສະດີກ່ຽວກັບການສ້າງຖານຂໍ້ມູນ ແລະ ຄວາມຮູ້ກ່ຽວກັບພາສາໃນການພັດທະນາລະບົບ. ລາຍລະອຽດຂອງແຕ່ລະທິດສະດີແມ່ນຈະໄດ້ນຳສະເໜີໂດຍສັງເຂບດັ່ງຕໍ່ໄປນີ້:

* + 1. ທິດສະດີກ່ຽວກັບການວິເຄາະ ແລະ ການອອກແບບລະບົບ

ການ​ວິ​ເຄາະ​ລະ​ບົບ​ເປັນ​ຂະ​ບວນ​ການ​ທຳ​ຄວາມ​ເຂົ້າ​ໃຈ ແລະ ກຳ​ນົດ​ລາຍ​ລະ​ອຽດ​ຂອງ​ບັນ​ຫາ​ເພື່ອ​ຈະ​ໄດ້​ພິ​ຈາ​ລະ​ນາ​ນຳ​ເອົາ​ລະ​ບົບ​ຂໍ້​ມູນ​ຂ່າວ​ສານ​ໃດ​ເຂົ້າ​ໄປ​ແກ້​ບັນ​ຫາ​ເລົ່າ​ນັ້ນ, ສ່ວນ​ການ​ອອກ​ແບບ​ລະ​ບົບ​ໝາຍ​ເຖິງ​ຂະ​ບວນ​ການ​ກຳ​ນົດ​ລາຍ​ລະ​ອຽດ​ຕ່າງໆ​ວ່າ​ຈະ​ຕ້ອງ​ເຮັດ​ແນວ​ໃດ​ກັບ​ອົງ​ປະ​ກອບ​ຂອງ​ລະ​ບົບ​ຂໍ້​ມູນ​ຂ່າວ​ສານ​ເພື່ອ​ຈະ​ໄດ້​ນຳ​ໄປ​ໃຊ້​ໃຫ້​ເກີດ​ຜົນ​ໃນ​ທາງ​ພາຍນອກ​ໄດ້. ສິ່ງ​ທີ່​ນຳ​ສະ​ເໜີ​ໃນ​ຫົວ​ຂໍ້​ນີ້​ໄດ້​ຂັດ​ສັນ​ພາກ​ສ່ວນ​ໜຶ່ງ​ຈາກ​ (ພ​ຣະ ເຫຼົາ​ຄຳ ເພັດ​ວິ​ໄລ, ວິ​ລະ​ຍຸດ ວົງ​ທິ​ລາດ ແລະ ເພັດ​ດາວວອນ ທິ​ແກ້ວ, 2016) ແລະ (ສົມ​ມິດ ທຸມ​ມາ​ລີ ແລະ ອາ​ມອນ ຈັນ​ທະ​ພາ​ວົງ​, 2012).

* + - 1. ວົງຈອນພັດທະນາລະບົບ

ເປັນ​ຂະ​ບວນ​ການ​ທີ່​ສະ​ແດງ​ເຖິງ​ການ​ດຳ​ເນີນ​ຂັ້ນ​ຕອນ​ການ​ເຮັດ​ວຽກ​ຂອງ​ລະ​ບົບ​ຕັ້ງ​ແຕ່​ຕົ້ນ​ຈົນ​ຈົບ, ບົດ​ໂຄງ​ການ​ຈົບ​ຊັ້ນ​ນີ້​ພ​ວກ​ຂ້າ​ພະ​ເຈົ້າໄດ້​ນຳ​ໃຊ້​ທິດ​ສະ​ດີ​ການ​ວິ​ເຄາະ ແລະ ອອກ​ແບບ​ລະ​ບົບ​ແບບ​ໂຄງ​ທີ່​ປະ​ກອບ​ມີ 5 ໄລ​ຍະ​ຄື:

* ໄລຍະທີ່ 1 ການວາງແຜນໂຄງການ (Project Planning phase)
* ໄລຍະທີ່ 2 ການວິເຄາະ (Analysis phase)
* ໄລຍະທີ່ 3 ການອອກແບບ (Design phase)
* ໄລຍະທີ່ 4 ການນຳໄປໄຊ້ (Implementation phase)
* ໄລຍະທີ່ 5 ການບໍາລຸງຮັກສາ (Maintenance phase)

ກ. ໄລຍະທີ່ 1 ການວາງແຜນໂຄງການ (Project planning phase)

ການວາງແຜນໂຄງການຈັດເປັນຂະບວນການພື້ນຖານໃນຄວາມເຂົ້າໃຈຢ່າງເລິກເຊິ່ງວ່າ ເປັນຫຍັງຕ້ອງສ້າງລະບົບໃໝ່ ທິມງານຕ້ອງພິຈາລະນາວ່າ ຈະຕ້ອງດຳເນີນງານຕໍ່ໄປແນວໃດກ່ຽວກັບຂະບວນການສ້າງລະບົບໃໝ່ ກ່ອນອື່ນໝົດຕ້ອງມີຈຸດກຳເນີດຂອງລະບົບງານເຊິ່ງໂດຍປົກກະຕິແລ້ວຈຸດກຳເນີດຂອງລະບົບງານມັກເກີດຂຶ້ນຈາກຜູ້ໃຊ້ລະບົບເປັນຜູ້ທືີຕິດແທດກັບລະບົບໂດຍກົງເຮັດໃຫ້ມີຄວາມໃກ້ສິດກັບລະບົບງານທີ່ດຳເນີນຢູ່ຫຼາຍທີ່ສຸດເມື່ອຜູ້ໃຊ້ລະບົບມີຄວາມຕ້ອງການປັບປຸງລະບົບ ງານດັ່ງນັ້ນຈຶ່ງຖືເປັນຈຸດເລີ່ມຕົ້ນໃນບົດບາດຂອງນັກວິເຄາະລະບົບວ່າຈະຕ້ອງສຶກສາເຖິ່ງຂອບເຂດ ຂອງບັນຫາທີ່ຜູ້ໃຊ້ລະບົບກຳລັງປະສົບບັນຫາຢູ່ ແລະ ຈະດຳເນີນການແກ້ໄຂແນວໃດ ສຶກສາເຖິງຄວາມເປັນໄປໄດ້ວ່າລະບົບໃໝ່ທີ່ຈະພັດທະນາຂຶ້ນມານັ້ນມີຄວາມເປັນໄປໄດ້ ແລະ ຄຸ້ມຄ່າກັບການຈະລົງທຶນຫຼື່ບໍ່.

ແນວໃດກໍ່ຕາມໄລຍະຂອງການວາງແຜນໂຄງການປົກກະຕິມັກຈະມີໄລຍະເວລາທີ່ສັ້ນ ແຕ່ກໍ່ຖືວ່າເປັນຂັ້ນຕອນທີ່ສຳຄັນທີ່ຈະໃຫ້ເກີດຜົນສຳເລັດໄດ້ດັ່ງນັ້ນໃນໄລຍະຂອງການວາງແຜນໂຄງການຈຶ່ງໄດ້ອາໄສນັກວິເຄາະລະບົບທີ່ມີຄວາມຮູ້ ແລະ ປະສົບການສູງເນື່ອງຈາກວ່າຫາກນັກວິເຄາະລະບົບບໍ່ເຂົ້າໃຈເຖິ່ງບັນຫາອັນແທ້ຈິງທີ່ເກີດຂຶ້ນ ກໍ່ຈະບໍ່ສາມາດພັດທະນາລະບົບຂຶ້ນມາເພື່ອແກ້ໄຂບັນຫາໃຫ້ຖືກຈຸດໄດ້ ແລະ ມັກຈະມີໂຄງການພັດທະນາລະບົບຫຼາຍໂຄງການທີ່ຫຼັງຈາກໄດ້ດຳເນີນການພັດທະນາ ແລະ ນໍາມາໃຊ້ງານແລ້ວປະກົດວ່າບໍ່ສາມາດຕອບສະໜອງຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ງານເຊິ່ງຖືວ່າເປັນເລື່ອງທີ່ກໍ່ໃຫ້ເກີດຄວາມສູນເສຍທັງທາງດ້ານການລົງທຶນ ແລະ ໄລຍະເວລາ.

ສະຫຼຸບລວມແລ້ວໄລຍະຂອງການວາງແຜນໂຄງການປະກອບມີກິດຈະກໍາຕ່າງໆຕໍ່ໄປນີ້:

- ກໍານົດບັນຫາ Problem Definition

- ສຶກສາຄວາມເປັນໄປໄດ້ຂອງໂຄງການ Feasibility study

- ສ້າງຕາຕະລາງກໍານົດເວລາໂຄງການ Project scheduling

- ຈັດຕັ້ງທິມງານໂຄງການ Staff the project

ຂ. ໄລຍະທີ່ 2 ການວິເຄາະ (Analysis Phase)

ໄລຍະການວິເຄາະຈະຕ້ອງມີຄຳຕອບກ່ຽວກັບຄຳຖາມວ່າໃຜເປັນຜູ້ທີ່ໃຊ້ລະບົບ ແລະ ມີຫຍັງແດ່ທີ່ຈະຕ້ອງເຮັດໃນໄລຍະນີ້ນັກວິເຄາະລະບົບຈະຕ້ອງດຳເນີນການໃນຂັ້ນຕອນຂອງການວິເຄາະລະບົບງານປັດຈຸບັນ (Current system) ເພື່ອນໍາມາພັດທະນາແນວຄວາມຄິດສໍາລັບລະບົບໃໝ່ (New System).

ຈຸດປະສົງຫຼັກຂອງການວິເຄາະຄືຈະຕ້ອງສຶກສາ ແລະ ສ້າງຄວາມເຂົ້າໃຈໃນຄວາມຕ້ອງການຕ່າງໆທີ່ໄດ້ລວບລວມມາດັ່ງນັ້ນການລວບລວມຄວາມຕ້ອງການ (Requirements Gathering) ຈຶ່ງຈັດເປັນງານສ່ວນພື້ນຖານຂອງການວິເຄາະລະບົບໂດຍຂໍ້ມູນຄວາມຕ້ອງການເຫຼົ່ານີ້ນັກວິເຄາະລະບົບຈະນຳມາວິເຄາະເພື່ອທີ່ຈະປະເມີນວ່າຄວນມີຫຍັງແນ່ທີ່ລະບົບໃໝ່ຕ້ອງດຳເນີນການ ແລະ ດ້ວຍເຫດນີ້ເອງການກຳນົດລາຍລະອຽດກ່ຽວກັບຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ (User Requirements) ຈະທະວີຄວາມສຳຄັນຫຼາຍຂຶ້ນເປັນລຳດັບສຳລັບລະບົບທີ່ມີຄວາມສັບຊ້ອນສູງ ແລະ ນັກວິເຄາະຕ້ອງເອົາໃນໃສ່ກັບການລວບລວມຄວາມຕ້ອງການຈາກຜູ້ໃຊ້ບໍ່ຄວນກຳນົດຄວາມຕ້ອງການຂຶ້ນເອງໂດຍໃຊ້ຄວາມຄິດສ່ວນຕົວຂອງຕົນເອງເປັນຫຼັກຫຼື່ປະເມີນຄວາມຕ້ອງການຂອງຜູ້ໃຊ້ລະບົບບໍ່ເຂົ້າກັບຈຸດປະສົງ ແລະ ຫາກມີການພັດທະນາລະບົບຕໍ່ໄປຈົນແລ້ວລະບົບງານທີ່ໄດ້ກໍ່ຈະບໍ່ກົງກັບຄາມຕ້ອງການຂອງຜູ້ໃຊ້ລະບົບຢ່າງແທ້ຈິງເຮັດໃຫ້ຕ້ອງມີຄວາມປັບປຸງຫຼືປ່ຽນແປງພາຍຫຼັງ.

ນັກວິເຄາະລະບົບສາມາດລວບລວມຄວາມຕ້ອງການຕ່າງໆໄດ້ຈາກການສັງເກດ, ການເຮັດວຽກຂອງຜູ້ໃຊ້, ການໃຊ້ເຕັກນິກ, ການສຳພາດຫຼື່ການໃຊ້ແບບສອບຖາມການອ່ານເອກະສານກ່ຽວກັບການປະຕິບັດງານຂອງລະບົບງານປັດຈຸບັນ, ລະບຽບກົດເກນຂອງບໍລີສັດ ແລະ ການມອບໝາຍຕຳ ແໜ່ງໜ້າທີ່ຮັບຜິດຊອບເຊິ່ງໃນຊ່ວງຂອງການເກັບກຳຂໍ້ມູນຄວາມຕ້ອງການກໍ່ຈະພົບກັບຜູ້ໃຊ້ໃນລະດັບຕ່າງໆທີ່ເຮັດໃຫ້ຮູ້ເຖິ່ງບັນຫາແລະແນວທາງການແກ້ໄຂບັນຫາທີ່ແນະນຳໂດຍຜູ້ໃຊ້ດັ່ງນັ້ນການເກັບກຳຄວາມຕ້ອງການຈຶ່ງເປັນກິດຈະກຳທີ່ສຳຄັນເພື່ອຄົ້ນຫາຄວາມຈິງແລະຕ້ອງສ້າງຄວາມຕ້ອງການເຊິ່ງກັນແລະກັນເພື່ອສະຫຼຸບອອກມາເປັນຂໍ້ກຳນົດທີ່ມີຄວາມຊັດເຈນໂດຍຂໍ້ກຳນົດເຫຼົ່ານີ້ເມື່ອຜູ້ໃຊ້ທີ່ກ່ຽວຂ້ອງໄດ້ອ່ານແລ້ວຈະຕ້ອງສຶກສາຄວາມໝາຍໄດ້ກົງກັນ.

ຫຼັງຈາກໄດ້ນໍາຄວາມຕ້ອງການຕ່າງໆມາສະຫຼຸບເປັນຂໍ້ກໍານົດທີ່ຊັດເຈນແລ້ວຂັ້ນຕອນຕໍ່ໄປກໍ່ຄືນໍາຂໍ້ກໍານົດເຫຼົ່ານັ້ນໄປພັດທະນາອອກມາເປັນຄວາມຕ້ອງການຂອງລະບົບໃໝ່ໂດຍເຕັກນິກທີ່ໃຊ້ກໍ່ຄືການພັດທະນາແບບຈໍາລອງຂະບວນການ (Process Model) ເຊິ່ງເປັນແຜນພາບທີ່ໃຊ້ອະທິບາຍເຖິງຂະບວນການທີ່ຕ້ອງດຳເນີນໃນລະບົບວ່າມີຫຍັງແດ່ແລະຕໍ່ໄປກໍ່ດໍຳເນີນການພັດທະນາແບບຈຳລອງຂໍ້ມູນ (Data Model) ເພື່ອອະທິບາຍເຖິງຂໍ້ມູນທີ່ຈັດເກັບໄວ້ສໍາລັບສະໜັບສະໜູນການເຮັກວຽກຕ່າງໆ.

ສະຫຼຸບໄລຍະຂອງການວິເຄາະລະບົບປະກອບມີກິດຈະກໍາຕ່າງໆດັ່ງນີ້;

- ວິເຄາະລະບົບງານປັດຈຸບັນ.

- ເກັບກໍາຄວາມຕ້ອງການໃນດ້ານຕ່າງໆແລະນໍາມາວິເຄາະເພື່ອສະຫຼຸບເປັນຂໍ້ກໍານົດຊັດເຈນ

- ນຳຂໍ້ກໍານົດມາພັດທະນາອອກມາເປັນຄວາມຕ້ອງການຂອງລະບົບໃໝ່.

- ສ້າງແບບຈຳລອງຂະບວນການຂອງລະບົບໃໝ່ໂດຍການແຕ້ມແຜນພາບກະແສຂໍ້ມູ (DataFlowDiagram:DFD).

- ສ້າງແບບຈໍາລອງຂໍ້ມູນໂດຍການແຕ້ມ Entity Relationship Deagram:ERD.

ຄ. ໄລຍະທີ່ 3 ການອອກແບບ (Design Phase)

ໄລຍະການອອກແບບເປັນການພິຈາລະນາວ່າລະບົບລະດໍາເນີນການໄປໄດ້ແນວໃດເຊິ່ງກ່ຽວ ຂ້ອງກັບຍຸດທະວິທີການຂອງການອອກແບບທີ່ວ່າດ້ວຍການຕັດສິນໃຈວ່າຈະພັດທະນາລະບົບໃໝ່ດ້ວຍ ແນວທາງໃດເຊັ່ນພັດທະນາຂຶ້ນເອງ,ຊື້ໂປຣແກມສຳເລັດຮູບຫຼືວ່າຈ້າງບໍລິສັດພັດທະນາລະບົບໃຫ້ເປັນ ຕົ້ນ. ນອກຈາກນີ້ໄລ່ຍະການອອກແບບຈະກ່ຽວຂ້ອງກັບການອອກແບບທາງດ້ານສະຖາປັດຕະຍະກຳລະບົບທີ່ກ່ຽວຂ້ອງກັບອຸປະກອນຮາດແວ,ຊອບແວແລະເຄືອຂ່າຍ. ການອອກແບບລາຍງານ (Out Design) ການອອກແບບໜ້າຈໍເພື່ອປະຕິສໍາພັນກັບຜູ້ໃຊ້ (User Interface), ການອອກແບບຜັງງານລະບົບ (System Flowchart), ເຊິ່ງລວມເຖິ່ງລາຍລະອຽດຂອງໂປຣແກມ (SpecificProgram), ຖານຂໍ້ມູນ (Database) ແລະຂໍ້ມູນທີ່ກ່ຽວຂ້ອງແນວໃດກໍ່ຕາມເຖິ່ງວ່າກິດຈະກຳບາງສ່ວນຂອງໄລຍະອອກແບບນີ້ສ່ວນໃຫ່ຍຈະຖືກດຳເນີນການໄປບາງສ່ວນແລ້ວໃນໄລຍະຂອງການວິເຄາະແຕ່ໄລຍະການອອກແບບນີ້ຈະເນັ້ນເຖິ່ງການດຳເນີນການແກ້ໄຂບັນຫາແນວໃດຫຼາຍກ່ວາໂດຍການນຳຜົນຂອງແບບຈຳລອງທາງ Logical Model. ທີ່ໄດ້ຈາກການວິເຄາະມາພັດທະນາມາເປັນແບບຈໍາລອງທາງ Physical Model.

- ການວິເຄາະຈະເນັ້ນແກ້ໄຂບັນຫາຫຍັງແດ່.

- ການອອກແບບຈະເນັ້ນການແກ້ໄຂບັນຫາແນວໃດ.

- ສະຫຼຸບໄລຍະການອອກແບບ.

- ພິຈາລະນາແນວທາງໃນການພັດທະນາລະບົບ.

- ອອກແບບສະຖາປັດຕະຍະກໍາລະບົບ (Architecture Design).

- ອອກແບບຖານຂໍ້ມູນ (Database Design).

- ອອກແບບການສະແດງຜົນ (Output Design).

- ອອກແບບການປ້ອນຂໍ້ມູນ (Input Design).

- ອອກແບບສ່ວນຕິດຕໍໍກັບຜູ້ໃຊ້ (User Interface).

- ສ້າງຕົ້ນແບບ (Prototype).

- ອອກແບບໂປຣແກມ (Structure Chart).

ງ. ໄລຍະທີ່ 4 ການນຳໄປໃຊ້ (Implementation Phase)

ໃນໄລຍະການນນໍາໄປໃຊ້ຈະເຮັດໃຫ້ລະບົບເກີດຜົນຂຶ້ນມາໂດຍການສ້າງລະບົບທົດສອບລະບົບແລະການຕິດຕັ້ງລະບົບໂດຍຈຸດປະສົງຫຼັກຂອງກິດຈະກຳໃນໄລຍະນີ້ບໍ່ແມ່ນພຽງຄວາມໜ້າເຊື່ອຖືຂອງລະບົບຫຼື່ລະບົບສາມາດເຮັດວຽກໄດ້ດີເທົ່ານັ້ນແຕ່ຕ້ອງໝັ້ນໃຈວ່າຜູ້ໃຊ້ລະບົບຕ້ອງໄດ້ຮັບການເຝິກອົບຮົມເພື່ອໃຊ້ງານລະບົບແລະຄວາມຄາດຫວັງໃນອົງກອນທີ່ຕ້ອງການຜົນຕອບແທນໃນດ້ານດີກັບການໃຊ້ລະບົບໃໝ່ລຳດັບກິດຈະກຳຕ່າງໆທຸກກິດຈະກຳຕ້ອງເຂົ້າມາດຳເນີນການຮ່ວມກັນໃນລະໄຍນີ້ເພື່ອໃຫ້ລະບົບການປະຕິບັດງານໄດ້ຮັບຄວາມປະສົບຜົນສຳເລັດໄດ້ໂດຍດີ.

ສະຫຼຸບໄລຍະການນໍາໄປໃຊ້ຈະປະກອບມີກິດຈະກໍາຕ່າງໆດັ່ງຕໍ່ໄປນີ້:

- ສ້າງລະບົບຂຶ້ນມາດ້ວຍການຂຽນໂປຣແກມ.

- ກວດສອບຄວາມຖືກຕ້ອງທາງດ້ານ Verification ແລະ Validation ແລະດໍາເນີນການທົດສອບລະບົບ.

- ແປງຂໍ້ມູນ (Convert Data).

- ຕິດຕັ້ງລະບົບ (System Installation) ແລະສ້າງເອກກະສານຄູ່ມື.

- ຝຶກອົບຮົມຜູ້ໃຊ້ ແລະ ປະເມີນຜົນລະບົບໃໝ່.

ສໍາລັບການສ້າງລະບົບຫຼື່ການຂຽນໂປຣແກມນັ້ນສາມາດໃຊ້ວິທີການຂຽນໂປຣແກມດ້ວຍພາສາຄອມພິວເຕີເຊັ່ນ: ການໃຊ້ພາສາ Visual Basic, Delphi, Java.ນອກຈາກນີ້ຫຍັງມີເຕັກນິກອື່ນໆເຊັ່ນ: ເຄື່ອງມືໃນການພັດທະນາ Application ເຊິ່ງເປັນຊອບແວທີ່ເປັນແຫຼ່ງລວມຂອງເຄື່ອງມືຕ່າງໆທີ່ໃຊ້ເພື່ອພັດທະນາ Application ເຮັດໃຫ້ຜູ້ຂຽນໂປຣແກມບໍ່ເຮັດວຽກໜັກຄືເມື່ອກ່ອນມີແຕ່ຮຽນຮູ້ແລະປະຍຸກໃຊ້ເຄື່ອງມືເຫຼົ່ານັ້ນກໍ່ສາມາດພັດທະນາລະບົບງ່າຍຂຶ້ນ.

ຈ. ໄລຍະທີ່ 5 ການບໍາລຸງຮັກສາ (Maintenance Phase)

ໂດຍປົກກະຕິແລ້ວໄລຍະການບໍາລຸງຮັກສາຈະບໍ່ນໍາເຂົ້າໄປລວມໃນສ່ວນຂອງ SDLC ຈົນກະ ທັ້ງລະບົບມີການຕິດຕັ້ງເພື່ອໃຊ້ງານແລ້ວເທົ່ານັ້ນໄລຍະນີ້ຈະໃຊ້ເວລາຍາວນານທີ່ສຸດເມື່ອທຽບກັບໄລ ຍະອື່ນໆທີ່ຜ່ານມາເນື່ອງຈາກລະບົບຈະຕ້ອງໄດ້ຮັບການບຳລຸງຮັກສາຕະຫຼອດໄລຍະເວລາທີ່ມີການໃຊ້ລະບົບສິ່ງທີ່ຄາດຫວັງຂອງໜ່ວຍງານກໍ່ຄືຕ້ອງການໃຫ້ລະບົບໃຊ້ງານຍາວນານຫຼາຍປີລະບົບສາມາດຮອງຮັບເຕັກໂນໂລຊີໃໝ່ໆໃນອະນາຄົດໄດ້ດັ່ງໜັ້ນໃນຊ່ວງໄລຍະເວລາດັ່ງກາວຈຶ່ງສາມາດເພີ່ມເຕີ່ມຄວາມສາມາດຂອງລະບົບໃຫ້ມີປະສິດຕິພາບສູງຂຶ້ນພ້ອມທັງການແກ້ໄຂປັບປຸງໂປຣແກມໃນກໍລະນີທີ່ເຫັນຂໍ້ຜິດພາດ.

- ການບໍາລຸງຮັກສາລະບົບ (System Maintenannce).

- ການເພີ່ມເຕີມຄວາມສາມາດໃໝ່ໆເຂົ້າໃນລະບົບ (Enhance System).

- ສະໜັບສະໜູນງານຂອງຜູ້ໃຊ້ (Support the User).

ຈາກໄລຍະຕ່າງໆຕາມຂັ້ນຕອນການພັດທະນາລະບົບຕາມແບບແຜນຂອງ SDLC ຈະເຫັນວ່າມີການໃຊ້ຄໍາວ່າໄລຍະແລະກິດຈະກໍາເຊິ່ງສາມາດອະທິບາຍລາຍລະອຽດເພື່ອໃຫ້ເກີດຄວາມເຂົ້າໃຈກົງກັນດັ່ງນີ້:

- ໄລຍະ(Phase) ຄືກຸ່ມກິດຈະກໍາທີ່ກ່ຽວຂ້ອງກັນ.

- ກິດຈະກໍາ (Activity) ຄືກຸ່ມຂອງງານທີ່ກຽວຂ້ອງກັນ.

- ໜ້າວຽກ (Task) ຄືງານທີ່ດໍາເນີນການເຊິ່ງຖືເປັນງານທີ່ນ້ອຍທີ່ສຸດ.

* + - 1. ແຜນວາດການໄຫຼຂໍ້ມູນ (Data Flow Diagram DFD)

- ເປັນແຜນພາບທີ່ສະຫຼຸບລວມຂໍ້ມູນທັງໝົດໄດ້ຈາກການວິເຄາະໃນລັກສະນະຂອງຮູບແບບທີ່ເປັນໂຄງສ້າງ.

- ເປັນຂໍ້ຕົກລົງຮ່ວມກັນລະຫວ່າງນັກວິເຄາະລະບົບ ແລະ ຜູ້ຊົມໃຊ້.

- ເປັນແຜນພາບທີ່ໃຊ້ໃນການພັດທະນາຕໍ່ໃນຂັ້ນຕອນຂອງການອອກແບບ.

- ຮູ້ທີ່ໄປທີ່ມາຂອງຂໍ້ມູນທີ່ໄຫຼໃນຂະບວນການຕ່າງໆ.

* + - 1. ສັນຍະລັກທີ່ໃຊ້ໃນແຜນວາດຂໍ້ມູນ

ຕາຕະລາງ ສະແດງສັນຍາລັກ **Data Flow Diagram**

|  |  |  |
| --- | --- | --- |
| ຊື່ | ສັນຍາລັກ | ຄວາມຫມາຍ |
| Boundary Or External Entity |  | ຂອບເຂດຫມາຍເຖິງພາກສ່ວນທີ່ກ່ຽວຂ້ອງເກັບລະບົບເຊິ່ງລະບົບບໍ່ສາມາດຄວບຄຸມໄດ້ |
| Process |  | ປະມວນຜົນຫຼືຫນ້າວຽກທີ່ເຮັດໃນໂຄງການນັ້ນໆ |
| Data Store |  | ບ່ອນຈັດເກັບຂໍ້ມູນ |
| Data Flow |  | ການໄຫຼຂອງຂໍ້ມູນ |
| Real-Time Link |  | ການເຊື່ອມໂຍງແບບໄກທີ່ມີການຕອບກັບແບບທັນທີທັນໃດ |

* + - 1. ຫຼັກການຂຽນແຜນວາດການໄຫຼຂໍ້ມູນ

1. Process:
   * ເມື່ອມີຂໍ້ມູນເຂົ້າໄປທີ່ Process ກໍ່ຕ້ອງມີຂໍ້ມູນ ຫຼື ຜົນຮັບອອກມາຈາກ Process ເຊັ່ນກັນຈະເປັນໄປບໍ່ໄດ້ທີ່ມີສະເພາະຂໍ້ມູນເຂົາຢ່າງດຽວ.
2. Data store:

* ຂໍ້ມູນຈະໄຫຼຈາກ Data store ໜຶ່ງໄປຫາ Data store ໜຶ່ງໂດຍກົງບໍ່ໄດ້ຈະຕ້ອງຜ່ານ Process ເທົ່ານັ້ນ
* ຂໍ້ມູນທີ່ສົ່ງຜ່ານ External entityບໍ່ສາມາດໄຫຼເຂົ້າໄປ Data storeໂດຍກົງໄດ້ຈະຕ້ອງໃຊ້ Processເປັນຕົວກາງໃນການເຊື່ອມໂຍງເພື່ອຈັດເກັບຂໍ້ມູນໃນ Data store
* ຂໍ້ມູນທີ່ໄຫຼຜ່ານຈາກ Data storeບໍ່ສາມາດເຊື່ອໂຍງເຂົ້າກັບ External entity ໄດ້ໂດຍກົງຈະຕ້ອງຜ່ານ Processເທົ່ານັ້ນ.

1. External entity:

* External entity ບໍ່ສາມາດເຊື່ອມໂຍງເຂົ້າຫາກັນໄດ້ຈະຕ້ອງໃຊ້ Process ເປັນຕົວກາງເພື່ອສົ່ງຜ່ານ ແລະ ຊື່ຂອງ External entity ຈະໃຊ້ຄໍານາມເທົ່ານັ້ນ.

1. Data flow:

* ການໄຫຼຂໍ້ມູນທີ່ມີຫົວຊີ້ໄປທີ່ Processໝາຍເຖິງ Process ມີການອ່ານ ຫຼື ການດືງຂໍ້ມູນຈາກ Data store ມາໃຊ້ວຽກ
* ການໄຫຼຂໍ້ມູນຈາກ Process ທີ່ມີຫົວລູກສອນຊີ້ໄປຍັງ Data store ໝາຍເຖິງການ Update ຫຼື ການເພີ່ມຂໍ້ມູນລົງໄປທີ່ Data store

|  |  |
| --- | --- |
| ອະນຸຍາດ | ບໍ່ອະນຸຍາດ |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* + - 1. Flowchart

ສັນຍະລັກ Flowchart ຄື ຮູບພາບທີ່ໃຊ້ແທນຄວາມຫມາຍການເຮັດວຽກງານໃນລັກສະນະຕ່າງໆ ພາຍໃນແຜນຜັງ (Flowchart) ປະກອບໄປດ້ວຍ ການເລີ່ມຕົ້ນ (Start), ການຈົບ (End), ການກະທຳ (Process), ການນຳເຂົ້າຂໍ້ມູນ (Input), ການສະແດງຜົນຂໍ້ມູນ (Output), ການຕັດສິນໃຈ (Decision), ຄຳອະທິບາຍ (Annotation), ຈຸດເຊື່ອມຕໍ່ (Connector), ທິດທາງການເຮັດວຽກງານ (Direction Flow)

ສັນຍະລັກເຫຼົ່ານີ້ເມື່ອຖືກນຳມາເຊື່ອມຕໍ່ກັນ ຈະກາຍເປັນ "ແຜນຜັງ (Flowchart)" ທີ່ສະແດງລຳດັບຂັ້ນຕອນການເຮັດວຽກງານເພື່ອ

* ເປັນ​ເຄື່ອງ​ມື​ໃນ​ການ​ຈັດ​ລຳ​ດັບ​ຄວາມ​ຄິດ
* ເຫັນ​ລຳ​ດັບ​ຂັ້ນ​ຕອນ​ການ​ເຮັດ​ວຽກງານ​ທີ່​ຊັດເຈນ

(ຄວາມໝາຍສັນຍະລັກ Flowchart ຄວາມໝາຍ ແລະ ວິທີໃຊ້ງານ <http://share.olanlab.com/th/it/blog/view/211>  Dec 01, 2017​).

ສັນຍະລັກ Flowchart

|  |  |
| --- | --- |
| ຮູບ​ພາບ​ສັນຍະລັກ | ຄວາມຫມາຍ​ຂອງ​ສັນຍະລັກ |
|  | ການ​ເລີ່ມ​ຕົ້ນ​ຫລື​ຈົບ Flowchart (Start ຫລື End) |
|  | ການ​ກະທຳ (Process) ຖືກ​ໃຊ້​ເພື່ອ​ສະແດງ​ທີ່​ການ​ກະທຳ​ໃນ Flowchart |
|  | ຮັບຂໍ້ມູນ |
|  | ການ​ຕັດ​ສິນ​ໃຈ (Decision) |
|  | ສະແດງຜົນທາງຈໍພາບ |
|  | ຈຸດ​ເຊື່ອມ​ຕໍ່ (Connector) |
|  | ທິດ​ທາງ​ການ​ເຮັດ​ວຽກງານ (Direction Flow) |

* + 1. ທິດສະດີກ່ຽວກັບຖານຂໍ້ມູນ

Database ຫຼື ຖານຂໍ້ມູນຄືກຸ່ມຂອງຂໍ້ມູນທີ່ຖືກເກັບລວບລວມໄວ້ໂດຍມີຄວາມສໍາພັນເຊິ່ງກັນ ແລະ ກັນ ໂດຍບໍ່ໄດ້ບັງຄັບວ່າຂໍ້ມູນທັງຫມົດນີ້ຈະຕ້ອງເກັບໄວ້ໃນແຟ້ມຂໍ້ມູນດຽວກັນ ຫຼື ແຍກເກັບຫຼາຍໆແຟ້ມຂໍ້ມູນ

ລະບົບຖານຂໍ້ມູນຄືລະບົບທີ່ລວບລວມຂໍ້ມູນຕ່າງໆທີ່ກ່ຽວຂ້ອງກັນເຂົ້າໄວ້ດ້ວຍກັນຢ່າງມີລະບົບ, ມີຄວາມສໍາພັນລະຫວ່າງຂໍ້ມູນຕ່າງໆທີ່ຊັດເຈັນໃນລະບົບຖານຂໍ້ມູນຈະປະກອບດ້ວຍແຟ້ມຂໍ້ມູນຫຼາຍແພ້ມທີ່ມີຂໍ້ມູນກ່ຽວຂ້ອງກັນ,ສໍາພັນກັນເຂົ້າໄວ້ດ້ວຍກັນຢ່າງເປັນລະບົບ ແລະ ເປີດໂອກາດໃຫ້ ຜູ້ໃຊ້ສາມາດໃຊ້ງານ ແລະ ຮັກສາປ້ອງກັນຂໍ້ມູນເຫຼົ່ານີ້ໄດ້ຢ່າງມີປະສິດທິພາບໂດຍມີຊອບແວຣ໌ທີ່ປຽບສະເຫມືອນຊື່ກາງລະຫວ່າງຜູ້ໃຊ້ ແລະໂປຣແກຣມຕ່າງໆທີ່ກ່ຽວຂ້ອງກັບການໃຊ້ຖານຂໍ້ມູນເອີ້ນວ່າລະບົບຈັດການຖານຂໍ້ມູນ ຫຼື DBMS

* + - 1. ການເຮັດ Normalization

Normalization ເປັນຫຼັກການໜຶ່ງທີີຜູູອອກແບບຖານຂໍ້ມູນຈະຕ້ອງນໍາມາໃຊ້ໃນການແປງຂໍ້ມູນທີີຢູູໃນຮູບແບບທີີຊໍໍາຊ້ອນໃຫ້ຢູູໃນຮູບແບບທີີງ່າຍຕໍ່ການນໍາໄປໃຊ້ງານ ແລະ ກໍ່ໃຫູເກີດບັນຫານູ້ອຍທີີສຸດ.

* ຈຸດປະສົງຂອງການເຮັດການເຮັດ Normalization
* ຫຼຸດຜ່ອນຄວມຊໍ້າຊ້ອນຂອງຂໍ້ມູນ ເມື່ອຫຼຸດຄວາມຊໍ້າຊ້ອນຈະເຮັດໃຫ້ຫຼຸດເນື້້ອທີ່ໃນການຈັດເກັບຂໍ້ມູນ
* ຫຼຸດບັນຫາຄວາມບໍ່ຖຶກຕ້ອງຂອງຂໍ້ມູນ ເມື່ອຂໍ້ມູນບໍ່ເກີດຄວາມຊໍ້າຊ້ອນ ເຮັດໃຫ້ການປັບປຸງຂໍ້ມູນສາມາດເຮັດໄດ້ຈາກແຫຼ່ງຂໍ້ມູນພຽງບ່ອນດຽວ.
* ຫຼຸດຄວາມຜິດພາດທີ່ເກີດຈາກການປັບປຸງຂໍ້ມູນ

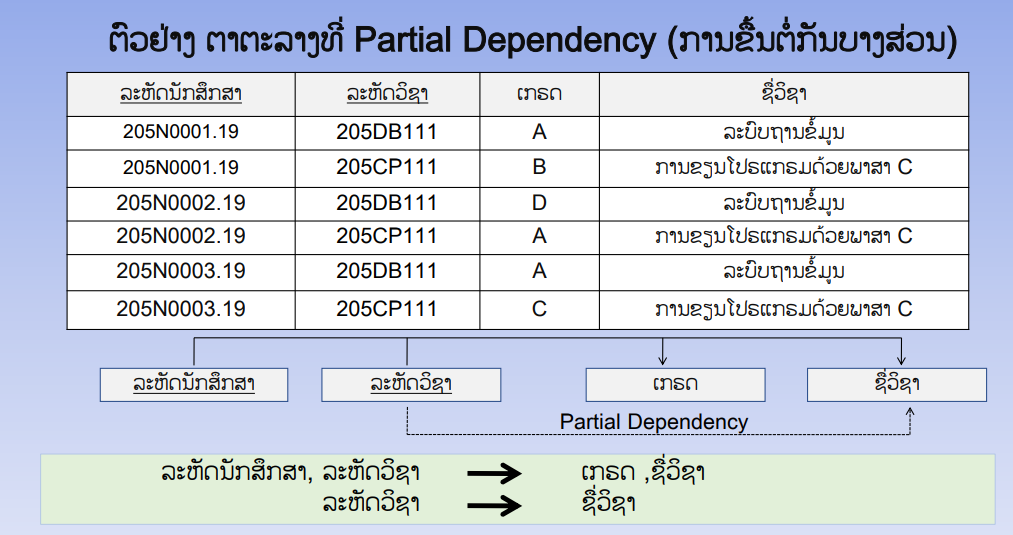
ກ. ຂັ້ນຕອນການເຮັດ Normalization

* Normalization ລະດັບ 1 ຫຼືເອີ້ນວ່າ 1NF
* Normalization ລະດັບ 2 ຫຼືເອີ້ນວ່າ 2NF
* Normalization ລະດັບ 3 ຫຼືເອີ້ນວ່າ 3NF

ຂ. First Normal Form (1NF)

* ທຸກ Attribute ໃນແຕ່ລະ record ຈະເປັນ single value ບໍ່ມີ ຄ່າຂອງກຸ່ມຂໍ້ມູນຊໍ້າຊ້ອນກັນ (Repeating Group).
* ຂໍ້ມູນທຸກແຖວ (Tuple) ຕ້ອງມີຄ່າບໍ່ຊໍ້າກັນ

ຄ. Second Normal Form (2NF)

* ຕ້ອງເປັນ First Normal Form (1NF) ມາກ່ອນ
* ຕ້ອງບໍ່ມີ Partial Dependency (ການຂຶ້ນຕໍ່ກັນບາງສ່ວນ)
* ສະຫຼຸບ: ການເຮັດນໍມາລາຍເຊຊັນ ລະດັບ2 (Second normal form : 2NF) ເປັນການເຮັດໃຫ້ແອດທິບິວທີີ່ບໍ່ຂຶ້ນກັບຄີຫຼັກອອກໄປເພື່ອໃຫ້ແອດທິບິວອື່ນທັງໝົດຂຶ້ນກັບສ່ວນທີີ່ເປັນຄີຫຼັກ

ງ. Third Normal Form (3NF)

* Relation ນັ້ນຈະຕ້ອງມີຄຸນສົມບັດ 2NF
* ຕ້ອງບໍ່ມີຄວາມສໍາພັນລະຫວ່າງ Non-key Attribute ຫຼື ບໍ່ມີ Transitive Dependency
* ຈ. ສະຫຼຸບ Normalization

• 1NF ທຸກແອດທິບິວໃນແຕ່ລະແຖວຕ້ອງມີຂໍ້ມູນພຽງຄ່າດຽວເທົ່ານັ້ນ

• 2NF ຣີເລເຊິນນັ້ນຕ້ອງບໍ່ມີຄວາມສໍາພັນລະຫວ່າງແອດທິບິວແບບບາງສ່ວນ(ແອດທິບິວ ທຸກຕົວຕ້ອງຂຶ້ນກັບຄີຫຼັກທຸກຕົວບໍ່ຂຶ້ນຢູູ່ກັບຕົວໃດຕົວໜຶ່ງ)

• 3NF ທຸກແອດທິບິວທີີ່ບໍ່ແມ່ນຄີຫຼັກບໍ່ມີຄຸນສົມບັດໃນການກໍານົດຄ່າຂອງແອດທິບິວຕົວອື່ນ

* + - 1. ແຜນວາດຄວາມສຳພັນ Entity (ER Diagram)

ການອອກແບບຈໍາລອງຖານຂໍ້ມູນ ER ນັ້ນກ່ອນອື່ນຕ້ອງຮູ້ຈັກຄວາມໝາຍສັນຍາລັກຕ່າງໆທີ່ໃຊ້ໃນການອອກແບບຈໍາລອງ ER ໄດ້ຢ່າງຖືກຕ້ອງດັ່ງນັ້ນໃນຫົວຂໍ້ນີ້ຈະເວົ້າເຖິງຄວາມໝາຍ ແລະ ການໃຊ້ງານສັນຍາລັກຕ່າງໆຂອງ ER.

* 1. ເອັນຕີຕີ້ (Entity)

ເອັນຕີຕີ້ຄືວັດຖຸທີ່ເຮົາສົນໃຈເຊິ່ງອາດເປັນໄດ້ທັງບຸກຄົນ, ສະຖານທີ່, ວັດຖຸ, ເຫດການ ຫຼື ແນວຄິດທີ່ກໍ່ໃຫ້ເກີດກຸ່ມຂອງຂໍ້ມູນທີ່ຕ້ອງການເອັນຕີຕີ້ແບ່ງອອກເປັນ 2 ປະເພດຄື:

* 1. Strong Entity:

ເປັນເອັນຕີຕີ້ທີ່ເກີດຂຶ້ນດ້ວຍຕົນເອງເປັນອິດສະຫຼະບໍ່ຂຶ້ນກັບເອັນຕີຕີໃດສັນຍາລັກທີ່ໃຊ້ຄືຮູບສີ່ຫຼ່ຽມ ແລະ ສາມາດເອີ້ນ Strong Entity ໄດ້ອີກຊື່ໜື່ງວ່າ Regular Entity.

CUSTOMER

STOCK

ຮູບທີ 1 ຮູບ Strong Entity

* 1. Weak Entity:

ເອັນຕີຕີ້ຊະນິດນີ້ຈະຂຶ້ນກັບເອັນຕີຕີ້ຊະນິດອື່ນໆບໍ່ສາມາດເກີດຂຶ້ນໄດ້ຕາມລໍາພັງ ແລະ ຈະຖືກລົບເມື່ອເອັນຕີຕີ້ຫຼັກຖືກລົບອອກ ສັນຍາລັກທີ່ໃຊ້ຄືຮູບສີ່ຫຼ່ຽມຊ້ອນກັນ.

CUSTOMER

STOCK

ຮູບທີ 2 ຮູບWeak Entity

* 1. ແອັດທຣິບີວ (Attribute)

ແອັດທີຣບີວຄືຄຸນສົມບັດຂອງສົມບັດຂອງເອັນຕີຕີ້ສັນຍາລັກຂອງແອັດທີຣບີວຈະເປັນຮູບວົງມົນແອັດທຣິບີວໃດທີ່ຖືກໃຊ້ເປັນຄີຫຼັກຈະຖືກຂີດເສັ້ນກ້ອງກໍາກັບໄວ້.

emName

Position

emNo

EMPLOYEE

ຮູບທີ 3 ຮູບAttribute

* 1. ຄວາມສໍາພັນ (Relation)

ຄວາມສໍາພັນໃນທີ່ນີ້ໝາຍເຖິງຄວາມສໍາພັນລະຫວ່າງເອັນຕີຕີ້ໂດຍແຕ່ລະຄວາມສໍາພັນຄວນມີຊື່ລະບຸໄວ້ເພື່ອໃຊ້ອະທິບາຍເຊິ່ງປົກກະຕິຈະໃຊ້ສັນຍາລັກຮູບດອກຈັນທີ່ພາຍໃນລະບຸຄໍາກິລິຍາໄວ້ເພື່ອອະທິບາຍຄວາມສໍາພັນ.

SUBJECT

teaches

LUCTURE

M

1

ຮູບທີ 4 ຮູບAttribute

1. ຄວາມສໍາພັນແບບຢູນາຣີ(Unary Relationships):ເປັນຄວາມສໍາພັນລະຫວ່າງເອັນຕີຕີ້

ດຽວໂດຍຄວາມສໍາພັນແບບຢູນາຣີນີ້ກໍ່ຄືຄວາມສໍາພັນແບບຣີເຄີຊີບ (Recursive)ທີ່ເອີ້ນໃຊ້ໃນຕົວ

ນັ້ນເອງ.

manages

EMPLOYEE

1

M

ຮູບທີ 5 ຮູບAttribute

1. ຄວາມສໍາພັນແບບໄບນາຣີ (Binary Relationships):ການສໍາພັນຊະນິດນີ້ຈະມີເອັນຕີຕີ້ທີ່ກ່ຽວຂ້ອງກັນ2ເອັນຕີຕີ້ເຊິ່ງເປັນຄວາມສໍາພັນຊະນິດຫນຶ່ງທີ່ສາມາດພົບເຫັນໄດ້ຫຼາຍໃນຄວາມເປັນຈິງ.

MAJOR

live

FACULTY

M

1

ຮູບທີ 6 ຄວາມສໍາພັນແບບໄບນາຣີ *( Binary Relationships* )

1. ຄວາມສໍາພັນແບບເທີ້ນາຣີ (Ternary Relationships) ເປັນຄວາມສໍາພັນທີມກ່ຽວຂ້ອງ 3 ເອັນຕີຕີ້ດ້ວຍກັນ.

1

M

PATIENT

DOCTOR

write

Read

PRESCRIPTION

Appears in

output

ຮູບທີ 7 ຄວາມສໍາພັນແບບເທີ້ນາຣີ (Ternary Relationships)

* + - 1. ພາສາ NoSql (Not Only SQL)

  ຫຼາຍ​ຄົນ​ຄົງ​ພໍ​ຈະ​ເຄີຍ​ໄດ້ຍິນ​ກັນ​ມາແດ່ກ່ຽວກັບ​ເທັກ​ໂນ​ໂລ​ຢີ​ການ​ຈັດການ​ຂໍ້​ມູນ​ແບບ​ໃໝ່​ນີ້ ຊຶ່ງ​ກໍ​ຄື NoSQL ເມື່ອ​ເວົ້າ​ເຖິງ NoSQL ຈະ​ໄດ້ຍິນ​ຊື່​ເວັບ​ໄຊ​​ທີ່​ໃຫ່ຍ​ໆ ເຊັ່ນ Facebook, Twitter, FourSquare, Digg ແລະ ​ອື່ນ​ໆ.  ເຮັດໃຫ້​ເຮົາ​ຮັບ​ຮູ້​ວ່າ NoSQL ເປັນ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ສຳລັບ​ງານ​ທີ່​ຕ້ອງ​ຮອງ​ຮັບ​ຂໍ້​ມູນ​ຂະໜາດ​ໃຫ່ຍໆຮອງ​ຮັບ​ການ​ຂະຫຍາຍ​ລະບົບ​ໄດ້​ງ່າຍເປັນ​ຕົ້ນ.

ຊຶ່ງ​ກໍ​ເປັນ​ເຊັ່ນ​ນັ້ນ​ແທ້ ແຕ່​ງານ​ທີ່​ນ້ອຍ​ໆຈະ​ເຮັດ​ຢ່າງໃດໃຊ້​ງານ​ໄດ້​ບໍ່ ຄຸ້ມ​ຄ່າ​ທີ່​ຈະ​ນຳ NoSQL ມາ​ໃຊ້​ງານ​ຂະໜາດ​ນ້ອຍ​ ຫຼື ບໍ່ ຫຼື​ໃຊ້ Relational Database ກໍ​ພຽງ​ພໍ​ແລ້ວ ຄຳ​ຕອບ​ຄືຂຶ້ນ​ຢູ່​ກັບ​ລັກສະນະ​ໃນ​ການ​ນຳ​ມາ​ໃຊ້​ງານ ກ່ອນທີ່ຈະ​ຕອບ​ຄຳ​ຖາມວ່າ NoSQL ເປັນ​ຄຳ​ຕອບ​ຂອງ​ລະບົບ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຂອງ​ຫຼືບໍ່ ລອງ​ພິຈາລະນາ​ຫົວຂໍ້​ຕ່າງ​ໆດັ່ງ​ຕໍ່ໄປ​ນີ້

1. ຜູ້ໃຊ້ນັບມື້ນັບຫຼາຍ (BigUsers)

ຈະ​ເຫັນ​ໄດ້​ວ່າ​ໃນ​ຊ່ວງ​ເວລາ​ທີ່​ຜ່ານມາ​ນີ້ ແລະ ​ໃນ​ປັດ​ຈຸ​ບັນຜູ້​ທີ່​ໃຊ້​ງານ Internet ມີ​ແນວ​ໂນ້ມ​ຫລາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ບໍ່​ວ່າ​ຈະ​ໃຊ້​ງານ​ຜ່ານ Desktop PC ຫລື Smartphone ຊຶ່ງ​ເທັກ​ໂນ​ໂລ​ຊີ​ຂອງ​ອຸປະ ກອນ (Devices) ມີ​​ຫຼາຍ​​ຂຶ້ນ ແລະ ​ໃຊ້​ງານ​ໄດ້​ງ່າຍ​ຂຶ້ນ

ການ​ພັດທະນາ​ລະບົບ​ໃຫ້​ສາມາດ​ຮອງ​ຮັບ​ປະລິມານ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ແຕ່​ລະ​ອຸປະກອນ (Devices) ເປັນ​ສິ່ງ​ໜຶ່ງ​ທີ່​ຕ້ອງ​ນຳ​ມາ​ພິຈາລະນາ  ແລະ ​ບໍ່​ພຽງ​ແຕ່​ຕ້ອງ​ຮອງ​ຮັບ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ຂອງ​ຜູ້​ໃຊ້​ໄດ້​ເທົ່າ​ນັ້ນ  ເຮົາ​ຕ້ອງ​ຮອງ​ຮັບ​ວິທີ​ການ​ປ້ອນ​ຂໍ້​ມູນ​ແບບ​ໃໝ່ຄືແຕ່​ກ່ອນ​ຜູ້​ຈັດການ​ເນື້ອ​ຫາ​ຕ່າງ​ໆ ຄື Web Master, Web Editor, ຜູ້​ເບິ່ງ​ແຍງ​ລະບົບເປັນ​ຕົ້ນ ແຕ່​ປະຈຸບັນຜູ້​ທີ່​ປ້ອນ​ຂໍ້​ມູນ​ຄືຜູ້​ໃຊ້​ບໍລິການ (users) ໂດຍ​ກົງຜ່ານ​ອຸປະກອນ (Devices) ຕ່າງ​ໆ​ທີ່​ມີ​ຫຼາກ​ຫຼາຍ ແລະ ​ການ​ປ້ອນ​ຂໍ້​ມູນ​ກໍ​ງ່າຍ​ກວ່າ​ແຕ່​ກ່ອນ​ອີກດ້ວຍ

ຍັງມີ​ປັດ​ໄຈ​ອື່ນ​ໆເຊັ່ນ ເທດ​ສະການ​ສຳຄັນ​ໆທີ່​ຄົນ​ຈະ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ຫລາຍ​ເປັນ​ພິເສດ ຫຼື ​ຜູ້​ໃຊ້​ງານ​ທີ່​ບໍ່​ແມ່ນ​ແຕ່​ປະເທດ​ເຮົາ​ເທົ່າ​ນັ້ນ ເພາະ​ໂລກ​ອິນ​ເຕີເນັດ​ເຖິງ​ກັນ ອາດຈະ​ຕ້ອງ​ເບິ່ງ​ວ່າ​ລະບົບ​ເຮົາ​ມີ​ຜູ້​ເຂົ້າ​ໃຊ້​ງານ​ຈາກ​ຕ່າງ​ປະເທດ ຫຼື ທົ່ວ​ໂລກ ​ຫຼືບໍ່​ອີກ​ດ້ວຍ

ດັ່ງ​ນັ້ນ​ເຮົາ​ຕ້ອງ​ກັບ​ມາ​​ທົບທວນ​ວິທີ​ການ​ຈັດການ​ຖານ​ຂໍ້​ມູນ​ແລ້ວ​ວ່າ  ບໍ່​ແມ່ນ​ແຕ່​ເຮັດໃຫ້​ຮອງ​ຮັບ​ກັບ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ຂອງ​ຜູ້​ໃຊ້​ບໍລິການ​ເທົ່າ​ນັ້ນ ແຕ່​ຕ້ອງ​ຮອງ​ຮັບ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ຫລາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ໄດ້​ອີກ

1. ປະເພດຂໍ້ມູນຕ່າງໆ ແລະ ຂໍ້ມູນທີ່ຕ້ອງການຈັດເກັບຫຼາຍຂຶ້ນເລື້ອຍໆ (BigData)

  ຈາກ​ຕົວ​ແປ​ຂອງ​ຜູ້​ໃຊ້​ງານ​ມີ​ຫລາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ອຸປະກອນ​ໃນ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ກໍ​ຫຼາກ​ຫຼາຍປະ​ເພດ​ຂອງ​ຂໍ້​ມູນ​ທີ່​ໄດ້​ຈາກ​ແຕ່​ລະ​ອຸປະກອນ​ກໍ​ຫຼາກ​ຫຼາຍ​ປະ​ເພດ ເຊັ່ນ ຂໍ້ຄວາມ, ຮູບ​ພາບ, ສຽງ, ວິ​ດີ​ໂອ, ຕຳແໜ່ງ​ສະຖານ​ທີ່ (GeoLocation) ແລະ ​ອື່ນ​ໆ ແລະ ​ການ​ປ້ອນ​ຂໍ້​ມູນ​ເຫຼົ່າ​ນີ້​ກໍ​ງ່າຍ​ແສນ​ງ່າຍ ເພາະ​ເທັກ​ໂນ​ໂລ​ຢີ​ຂອງ​ຮາ​ດ​ແວ​ຣ ແລະ ​ຊອບແວ​​ມີ​ການ​ພັດທະນາ​ຫລາຍ​ຂຶ້ນ​ເລື້ອຍ​ໆ ໃຊ້​ງານ​ງ່າຍ​ຂຶ້ນສະ​ດວກ​ຂຶ້ນວ່ອງໄວ​ຂຶ້ນ​ເລື້ອຍ​ໆ

    ດັ່ງ​ນັ້ນ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ລັ່ງໄຫຼ​ເຂົ້າ​ມາ​ຈາກ​ອຸປະກອນ​ຕ່າງ​ໆ​ເຫຼົ່າ​ນີ້  ເຮົາ​ອາດຈະ​ຕ້ອງ​ນຳ​ມາ​​ວິ​ເຄາະ​ພຶດຕິກຳ​ຂອງ​ຜູ້​ໃຊ້​ບໍລິການ ການ​ສົ່ງ​ເສີມ​ກາ​ນຕະຫຼາດ ເຮັດ​ຂໍ້​ມູນ​ການ​ຕັດ​ສິນ​ໃຈ​ຂອງ​ຜູ້​ບໍລິຫານ ຂໍ້​ມູນ​ລູກ​ຄ້າ​ສຳພັນ ແລະ ​ອື່ນ​ໆ​ອີກ​ຫລວງ​ຫລາຍ  ການ​ນຳ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ແບບ​ເດີມ (Relational Database) ອາດຈະ​ບໍ່​ເໝາະ​ກັບ​ລັກສະນະ​ງານ​ບາງຢ່າງ​ອີກ​ຕໍ່ໄປ

1. ເຕັກໂນໂລຢີຮາດແວໄດ້ປ່ຽນແປງ ລາຄາກໍ່ຖືກຫຼຸດລົງແຕ່ວ່າປະສິດທິພາບດີຂື້ນ (Cloud Computing).

ເຮົາ​ອາດຈະ​ເລີ່ມ​ໄດ້ຍິນ​ຄຳ​ວ່າ Cloud Technology  ຊຶ່ງ​ກໍ​ມີ​ຫຼາຍ​ປະ​ເພດແຕ່​ໃນ​ທີ່​ນີ້ເຮົາ​ຈະ​ເວົ້າ​ເຖິງ​ໃນ​ແງ່​ຂອງ​ການ​ນຳ​ມາ​ໃຊ້​ງານໂດຍ​ປະຈຸບັນ​ຖ້າ​ໃຜ​ເຄີຍ​ໄດ້​ລອງ​ໃຊ້​ງານ EC2 ຂອງ Amazon ມາແດ່​ແລ້ວຈະ​ຮູ້​ຈັກ​ເປັນ​ຢ່າງ​ດີ​ວ່າການ​ຈະ​ມີ​ເຄື່ອງ Server ແຮງ​ໆ​ຈັກເຄື່ອງ ເປັນ​ເລື່ອງ​ທີ່​ງ່າຍ​ຫລາຍ​ ຫຼື ການ​ຈະ​ມີ Server 10 ເຄື່ອງ ຫລື 20 ເຄື່ອງ ນຳ​ມາ​ຕໍ່​ເປັນ Database Cluster ນັ້ນ​ງ່າຍ​ຫຼາຍ​ ພຽງແຕ່​ຄລິກ​ສ້າງ Instance ບໍ່​ຈັກ​ເທື່ອ​ກໍ​ໄດ້ Server ມາ​ໃຊ້​ງານ​ແລ້ວ ແລະ​ ລາ​ຄາ​ກໍ​ຖືກ​ຫລາຍ​ໆ ຖ້າ​ເຮົາ​ບໍ່​ໃຊ້​ງານ​ແລ້ວ ກໍ​ຍົກ​ເລີກ​ການ​ໃຊ້​ງານ ​ແລະ​ ຄືນ​ກັບ​ໄປ​ໄດ້​ທັນ​ທີ ເມື່ອ​ທຽບ​ກັບ​ສະໄໝ​ກ່ອນ​ທີ່​ເຮົາ​ຕ້ອງ​ການ​ມີ Server ຈັກ 10 ເຄື່ອງ ຈະ​ຕ້ອງ​ລົງ​ທຶນ​ຊື້​ເຄື່ອງ​ມາ​ຫລາຍ ຖ້າ​ເຊົາ​ໃຊ້​ງານ​ແລ້ວຕ້ອງ​ແບກ​ຮັບ​ພາລະ​ເຄື່ອງ Server ເຫຼົ່າ​ນີ້​ໄວ້​ຊຶ່ງ​ເປັນ​ຕົ້ນ​ທຶນ​ທີ່​ແພງ​ຫລາຍ​

ຈາກ​ທີ່​ກ່າວ​ມາ​ຂ້າງ​ເທິງ ເຮົາ​ບໍ່​ໄດ້​ເນັ້ນ​ຂໍ້​ດີ​ຂອງ EC2 ແຕ່​ຢ່າງ​ໃດແຕ່​ກຳ​ລັງ​ຈະ​ເນັ້ນ​ວ່າພາບ ລວມ​ຂອງ​ການ​ໃຊ້​ງານ Server ເລີ່ມ​ປ່ຽນ​ໄປຄື ໃຊ້​ງານ​ໄດ້​ງ່າຍ​ຂຶ້ນ ລາ​ຄາ​ຖືກ​ລົງ ແຕ່​ປະ​ສິດ​ທິ​ພາບ​ດີ​ຂຶ້ນ ຊຶ່ງ​ເປັນ​ສິ່ງ​ທີ່​ສຳຄັນ​ໃນ​ການ​ນຳ​ມາ​ພິຈາລະນາ​ຂອງ​ເທກ​ໂນ​ໂລ​ຢີ​ດ້ານ​ຖານ​ຂໍ້​ມູນ​ຄື ຖ້າ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຖານ​ຂໍ້​ມູນ​ໃຫຍ່​ໆ ຫລື​ ຮອງ​ຮັບ​ຜູ້​ໃຊ້​ງານ​ໄດ້​ເປັນ​ຈຳນວນ​ຫລາຍ​ໆ ການ​ຂະຫຍາຍ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ເປັນ​ເລື່ອງ​ທີ່​ງ່າຍ​ຂຶ້ນ ຊຶ່ງ​ເຮັດ​ໄດ້​ໂດຍ​ການ​ເອົາ Server ມາ​ຕໍ່​ໆ​ກັນ​ອອກ​ໄປ ຫລື​ ເອີ້ນວ່າ​ການ​ຂະຫຍາຍ​ອອກ​ແນວ​ນອນ (Scale Out) ບໍ່​ແມ່ນ​ການ​ຂະຫຍາຍ​ລະບົບ​ຄືແຕ່​ກ່ອນ ຄື​ຂະຫຍາຍ​ອອກ​ແນວ​ຕັ້ງ (Scale Up) ແລະ​ ຕ້ອງ​ໃຊ້​ເຄື່ອງ Server ທີ່​ມີ​ປະ​ສິດ​ທິ​ພາບ​ສູງ​ໆ ຊຶ່ງ​ຈະ​ມີ​ຕົ້ນ​ທຶນ​ທີ່​ແພງ​ກວ່າ​ການ​ຂະຫຍາຍ​ແບບ​ແນວ​ນອນຫລາຍ​

     ດັ່ງ​ນັ້ນ​ການ​ຂະຫຍາຍ​ລະບົບ​ທີ່​ຢູ່​ເທິງ​ພື້ນ​ຖານ​ຂອງ NoSQL ຄື​ຮອງ​ຮັບ​ການ​ຂະຫຍາຍ​ລະ ບົບ​ແບບ​ແນວ​ນອນ (Scale Out) ຊຶ່ງ​ຈະ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ໄປ​ເກັບ​ທີ່​ເຄື່ອງ Server ຫຼາຍ​ໆ​ເຄື່ອງ ແລະ ​ໃຊ້​ເຄື່ອງ Server ທົ້ວ​ໆ​ໄປ​ທີ່​ເອີ້ນວ່າ (Commodity Server) ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ໃຊ້ Server ທີ່​ເອີ້ນວ່າ Enterprise Server ທີ່​ມີ​ລາ​ຄາ​ແພງ​ຕາມ Spec ທີ່​ສູງ​ຂຶ້ນ​ເລື້ອຍ​ໆ ແລະ​ ການ​ບໍລິຫານ​ຈັດການ​ກໍ​ຍາກ​ຂຶ້ນ​ອີກ​ດ້ວຍ

1. ບັນຫາ​ຂອງ Relational Database

    ຖ້າ​ໃຜ​ທີ່​ໃຊ້​ງານ Relation Database ທີ່​ຕ້ອງ​ການ​ຮອງ​ຮັບ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຂະໜາດ​ໃຫຍ່​ໆ ຄົງ​ຫຼີກລ້ຽງບໍ່ໄດ້ເລື່ອງ​ການເຮັດ Sharding ແລະ Distributed Cache ເພາະ​ເປັນ​ຕັວ​ຫຼັກ​ທີ່​ຕ້ອງ​ເຮັດ ເພື່ອ​ຂະຫຍາຍ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ຂອງ Relational Database ໃຫ້​ສາມາດ​ຮອງ​ຮັບ​ຂໍ້​ມູນ​ທີ່​ຫລາຍ​ຂຶ້ນ ແລະ​ຮອງ​ຮັບ​ຈຳນວນ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ລະບົບ​ໄດ້​ຫລາຍ​ຂຶ້ນ

1. Manual Sharding

    ການ​ແບ່ງ​ຕາ​ຕະລາງ​ຖານ​ຂໍ້​ມູນ (Table) ອອກ​ເປັນ​ສ່ວນ​ໆ ແລ້ວ​ກໍ່ການ​ກະ​ຈາຍ​ໄປ​ຈັດ​ເກັບ​ໃນ​ຫຼາຍ​ໆ Server  ເພື່ອ​ໃຫ້​ແຕ່​ລະ​ຕາ​ຕະລາງ (Table) ຂອງ​ຖານ​ຂໍ້​ມູນ​ບໍ່​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ຫຼາຍ​ເກີນ​ໄປ ເພາະ​ຖ້າ​ຂໍ້​ມູນ​ໃນ​ແຕ່​ລະ​ຖານ​ຂໍ້​ມູນ​ຫູາຍ​ເກີນ​ໄປ ຈະ​ເຮັດໃຫ້​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ຊ້າ​ ແຕ່​ບັນຫາ​ກໍ​ຈະ​ຕາມ​ມາ​ອີກ​ຄື ເມື່ອ​ຕ້ອງ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ອອກ​ໄປ​ໃນ​ແຕ່​ລະ Server ການ​ຈະ​ ​ກັບ​ຂໍ້​ມູນ ເຊັ່ນ ເພີ່ມ, ແກ້​ໄຂ, ລົບ, ດຶງ​ຂໍ້​ມູນ​ມາ​ສະແດງ​ຕ່າງ​ໆ ຈະ​ຕ້ອງ​ເຮັດ​ຜ່ານ Application ຫຼື ​ຕ້ອງ​ມີ Server ບາງ​ໂຕ​ທີ່​​ຖ້າ​ດຶງ​ຂໍ້​ມູນ​ແຕ່​ລະ Server ມາ​ທັງໝົດ​ເປັນ​ກ້ອນ​ດຽວ ນັ້ນ​ໝາຍ​ຄວາມ​ວ່າ ເຮົາ​ຕ້ອງ​ເຮັດ​ດ້ວຍ​ໂຕ​ເຮົາ​ເອງ ບໍ່​ແມ່ນ​ລະບົບ​ຖານ​ຂໍ້​ມູນ​ຈັດການ​ໃຫ້ (Manual Sharding)

1. Distributed Cache

    ເມື່ອ​ເຮົາ​ຕ້ອງ​ການ​ໃຫ້​ລະບົບ​ຮອງ​ຮັບ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ຫລາຍ​ໆ​ໄດ້​ນັ້ນ ຖ້າ​ຈະ​ຕ້ອງ​ເຂົ້າ​ມາ​ອ່ານ​ຂໍ້​ມູນ​ຜ່ານ Database ໂດຍ​ກົງ​ມັນອາດຈະ​ຮອງ​ຮັບ​ບໍ່​ໄຫວ ຫຼື ​ເຮັດ​ໄດ້​ຊ້າ ດັ່ງ​ນັ້ນ​ຈະ​ຕ້ອງ​ມີ​ການເຮັດ Cache Layer ຂຶ້ນ​ມາ ຄື​ແທນ​ທີ່​ຈະ​ເຂົ້າໄປ​ອ່ານ​ຈາກ​ຖານ​ຂໍ້​ມູນ​ໂດຍ​ກົງ ກໍ​ໃຫ້​ອ່ານ​ຜ່ານ Cache ກ່ອນ ດັ່ງ​ນັ້ນ​ການ​ອ່ານ​ຂໍ້​ມູນ​ຈາກ Cache ເປັນ​ການ​ອ່ານ​ຈາກ Memory ໂດຍ​ກົງ ເຮັດໃຫ້​ຮອງ​ຮັບ​ປະລິມານ​ການ​ເຂົ້າ​ມາ​ໃຊ້​ງານ​ໄດ້​ຫຼາຍ​ຂຶ້ນ

    ແຕ່​ບັນຫາ​ຄື ການເຮັດ Cache Layer ນີ້ ຮອງ​ຮັບ​ສະເພາະ​ການ​ອ່ານ​ຂໍ້​ມູນ​ເທົ່າ​ນັ້ນ ບໍ່​ຮອງ​ຮັບ​ການ​ຂຽນ​ຂໍ້​ມູນ​ໄດ້ ຖ້າ​ຕ້ອງ​ການ​ຮອງ​ຮັບ​ການ​ຂຽນ​ຂໍ້​ມູນ​ປະລິມານ​ຫຼາຍ​ໆ ແລະ​ອ່ານ​ຂໍ້​ມູນ​ປະລິມານ​ຫຼາຍ​ໆ ຈຶ່ງ​ເປັນ​ສິ່ງ​ທີ່ Relational Database ບໍ່​ສາມາດ​ຮອງ​ຮັບ​ງານ​ໃນ​ລັກສະນະ ອ່ານ​,​ຂຽນ ຂໍ້​ມູນ​ປະລິມານ​ຫຼາຍ​ໆໄດ້​ດີ​ ແລະ​ ສິ່ງ​ສຳຄັນການເຮັດ Cache Layer ຈະ​ຕ້ອງ​ມີ​ການ​ດູແລຮັກສາ ແລະ​ ໃຊ້ Server ແຍກ​ອອກ​ໄປ​ຕ່າງ​ຫາກ​ອີກ​

ຈາກ​ຈຸດ​ນີ້​ເອງ ທັງ​ການ​ເຮັດ Sharding ແລະ Caching ເປັນ​ສິ່ງ​ທີ່​ຖືກ​ພັດທະນາ​ຂຶ້ນ​ໃນ NoSQL ເທ​​ກ​ໂນ​ໂລ​​ຢີ  ໂດຍ​ຮອງ​ຮັບ Auto-Sharding ແລະ Integrated Caching ໃນ​​ຕັວ​ເອງ ດັ່ງນັ້ນ​ເຮົາ​ຈຶ່ງ​ໄດ້​ເຫັນ NoSQL ຖືກ​ນຳໄປ​ໃຊ້​ງານ​ກັບ​ລະບົບ​ໃຫຍ່ໆ ເຊັ່ນ Facebook, Twitter, FourSquare, Digg ແລະ​ອື່ນໆ ເພ​​າະ NoSQL ອອກ​ແບບ​ມາ​ເພື່ອ​ຮອງ​ຮັບ​ຄວາມ​ຕ້ອງການ​ງານ​ໃຫຍ່ໆ​ໄດ້​ດີ​ໂດຍ​ສະເພາະ​ຢູ່​ແລ້ວ ແຕ່​ເຖິງ​ຢ່າງ​ໃດກໍ​ຕາມ​ຍັງ​ມີ​ຄຸນສົມບັດ​ອື່ນໆ ທີ່​ໜ້າ​ສົນໃຈ​ໃນ NoSQL ເທ​​ກ​ໂນ​ໂລ​​ຢີ

ກ. ຄຸນສົມບັດ​ຂອງ NoSQL Database

* Dynamic Schemas

    ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຕ່າງ​ໆ ໃນ​ຖານ​ຂໍ້​ມູນ​ແບບ Relational Database ເຮົາ​ຈະ​ຕ້ອງ​ມີ​ການສ້າງ Schema ຫຼື ຮູບ​ແບບ​ຂອງ​ໂຄງ​ສ້າງ​ຕາ​ຕະລາງວ່າ​ຈະ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຫຍັງ​ ເມື່ອ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ເພິ່ມເຕີມ​ຕ້ອງ​ປ່ຽນ Schema ພາຍ​ຫຼັງ (Alter-Table) ກ່ອນ​ຈະ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຮູບ​ແບບ​ໃໝ່​ໄດ້

    ແຕ່​ໃນ​ປະຈຸບັນ ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ມີ​ການ​ປ່ຽນ​ແປງ​ຕະຫລອດ​ເວລາ ເພາະ​ຄວາມ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຕ່າງ​ໆ ມີ​ຫຼາກ​ຫຼາຍ​​ຂຶ້ນ​ເລື້ອຍ​ໆ ການ​ກຳນົດ​ໂຄງ​ສ້າງ​ຂອງ​ຕາ​ຕະລາງ​ຖານ​ຂໍ້​ມູນ ຫຼື ການ​ຕ້ອງ​ປ່ຽນ​ໂຄງ​ສ້າງ​ຖານ​ຂໍ້​ມູນ​ເລື້ອຍໆ ໂດຍ​ທີ່​ຂໍ້​ມູນ​ຍັງມີ​ຢູ່​ແລ້ວ ເປັນ​ເລື່ອງ​ທີ່​ຍາກ​ຫຼາຍ ​ຫຼື ເຮັດ​ບໍ່​ໄດ້​ເລີຍ ວິທີ​ການ​ຄື​ອາດ​ຕ້ອງ​ແຍກ​ອອກ​ເປັນ​ຕາ​ຕະລາງ​ໃໝ່ ຊຶ່ງ​ເປັນ​ວິທີ​ແກ້​ບັນຫາ​ຊົ່ວ​ຄາວ​ເທົ່າ​ນັ້ນ

    ລະບົບ​ຖານ​ຂໍ້​ມູນ​ແບບ NoSQL  ເຮົາ​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ມີ Schema ທີ່​ຕາຍ​ຕົວ ຫຼື ​ບໍ່​ຕ້ອງ​ມີ Schema ກ່ອນທີ່ຈະ​ຈັດ​ເກັບ​ຂໍ້​ມູນ ຂໍ້​ມູນ​ແຕ່​ລະ​ແຖວ ສາມາດ​ຈັດ​ເກັບ​ໄດ້​ຕາມ​ຕ້ອງ​ການ ຈະ​ເພີ່ມ​ ຫຼື ​ຫຼຸດ ກໍ​ບໍ່​ມີ​ບັນຫາ​ກັບ​ລະບົບ  ເຮັດໃຫ້​ເຮົາ​ສາມາດ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ໄດ້​ຕາມ​ທີ່​ຕ້ອງ​ການ ປ່ຽນ​ແປງ​ໄດ້​ຕະຫລອດ​ເວລາ ສະ​ດວກ​ ແລະ ວ່ອງໄວ

* Auto-Sharding

    ເມື່ອ​ຂໍ້​ມູນ​ມີ​ຂະໜາດ​ໃຫຍ່ ຫຼື ​ເຮົາ​ຕ້ອງ​ການ​ເພີ່ມ​ປະ​ສິດ​ທິ​ພາບ​ການ ອ່ານ​ ແລະ ຂຽນ​ຂໍ້​ມູນ​ປະລິ ມານ​ຫຼາຍ​ໆ ການເຮັດ Sharding ໃນ​ລະບົບ NoSQL Database ຈະ​ກໍ່ການ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ໄປ​ຍັງ Server ຕ່າງ​ໆ​ອັດ​ໂນ​ມັດ​​ (Auto-Sharding) ຜູ້​ພັດທະນາ (Developer) ບໍ່​ຕ້ອງ​ຂຽນ​ໂປຣ​ແກຣມ​ໃນ​ການ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ເອງຄືກັບ Relational Database, ການ​ກະ​ຈາຍ​ຂໍ້​ມູນ​ອອກ​ໄປ​ຫຼາຍ​ໆ Server ນີ້​ຍັງ​ເຮັດໃຫ້​ມີ​ຂໍ້​ດີ​ຄື ປະ​ຢັດ​ຕົ້ນ​ທຶນ​ໃນ​ການ​ຂະຫຍາຍ​ລະບົບ ເພາະ​ເປັນ​ການ​ຂະຫຍາຍ​ແບບ​ແນວ​ນອນ (Scale Out) ຊຶ່ງ​ສາມາດ​ນຳ Server ປົກກະຕິ​ທົ່ວ​ໄປ​ມາ​ໃຊ້​ງານ​ໄດ້ ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເປັນ Enterprise Server

* Replication

    ການ​ສຳ​ເນົາ​ຂໍ້​ມູນ​ຈາກ​ເຄື່ອງ​ໜຶ່ງ​ໄປ​ຢັງ​ອີກ​ເຄື່ອງ​ໜຶ່ງ (Replication) ເມື່ອ Server ໜຶ່ງ​ເສຍ​ຫາຍ ອີກ​ເຄື່ອງ​ໜຶ່ງ​ຈະ​ຂຶ້ນ​ມາ​ເຮັດວຽກ​ແທນ​ທັນ​ທີ ໂດຍ​ຂໍ້​ມູນ​ຂອງ​ແຕ່​ລະ​ເຄື່ອງ​ຈະ​ມີ​ຂໍ້​ມູນ​ຄືກັນ ດັ່ງ​ນັ້ນ Replication ເປັນ​ໜຶ່ງ​ຄຸນສົມບັດ​ທີ່​ຕອບສະໜອງ​ຕໍ່​ການ​ໃຊ້​ງານ​ທີ່​ຕ້ອງ​ການ​ຄວາມ​ຕໍ່​ເນື່ອງ​ໄດ້​ຕະຫລອດ​ເວລາ (High Availability)

* Integrated Caching

    ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ໃຊ້​ງານ​ເລື້ອຍໆ ເຂົ້າ​ໄວ້​ໃນ Memory (RAM) ຊຶ່ງ​ເປັນ​ຄຸນສົມບັດ​ເດັ່ນ​ຂອງ NoSQL ທີ່​ທັງຫມົດ Caching ໄວ້​ໃນ​ຕົວ​ເອງ​ຢູ່​ແລ້ວ ເຮົາ​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເຮັດ Cache Layer ຄືກັບ Relational Database ອີກ​ຕໍ່ໄປ ທີ່​ຕ້ອງ​ເຮັດ Cache Layer ແຍກ​ຕ່າງ​ຫາກ ​ແລະ​ ເບິ່ງແຍງຮັກສາ​ແຍກ​ອອກ​ໄປ​ຕ່າງ​ຫາກ​ອີກ​ດ້ວຍ

ຂ. ປະ​ເພດ​ຂອງ​ຖານ​ຂໍ້​ມູນ NoSQL

    NoSQL ຖືກ​ແບ່ງ​ປະ​ເພດ​ຕາມ​ລັກສະນະ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ແຕກ​ຕ່າງ​ກັນ ດັ່ງ​ນັ້ນ​ການ​ຈະ​ເລືອກ NoSQL Database ໂຕໃດໂຕໜຶ່ງ ຈະ​ຕ້ອງ​ເບິ່ງ​ອີກ​ວ່າ ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ຂອງ​ຖານ​ຂໍ້​ມູນ​ເປັນ​ແບບໃດ ເຊັ່ນ

- Document databases ເຊັ່ນ MongoDB, CouchDB, Elasticsearch

- Graph stores ເຊັ່ນ Neo4J, Infinite Graph, InfoGrid

- Key-value stores ເຊັ່ນ DynamoDB, Redis, MemcacheDB

- Wide-column stores ເຊັ່ນ Cassandra, Amazon SimpleDB, Hadoop / HBase

ຄ. Open source License

    ໂດຍ​ສ່ວນ​ໃຫຍ່​ແລ້ວ NoSQL ຈະ​ເປັນ​ລິຂະ​ສິດ​ແບບ Open source ຊຶ່ງ​ບໍ່​ຕ້ອງ​ເສຍ​ຄ່າ​ໃຊ້​ຈ່າຍ​ໃນ​ການ​ນຳ​ມາ​ໃຊ້​ງານ ດັ່ງ​ນັ້ນ​ເຮົາ​ສາມາດ​ນຳ NoSQL Database ແຕ່​ລະ​ຕົວ​ມາ​ຕິດ​ຕັ້ງ​ໃຊ້​ງານ​ໄດ້​ໂດຍ​ບໍ່​ເສຍ​ຄ່າ​ໃຊ້​ຈ່າຍ​ໃດ​ໆ (ຟ​ຣີ)

ງ. ນຳ NoSQL ມາ​ໃຊ້​ງານ​ຂະໜາດ​ນ້ອຍ​ໄດ້​ ຫຼື​ ບໍ່

    ຈາກ​ທີ່​ກ່າວ​ມາ​ແລ້ວ ຄົງ​ພໍ​ຈະ​ຕອບ​ຄຳ​ຖາມ​ນີ້​ໄດ້​ວ່າ ການ​ນຳ NoSQL Database ​ເມື່ອ​ນຳ​ມາ​ໃຊ້​ໃນ​ງານ​ຂະໜາດ​ໃຫຍ່​ນັ້ນ ເໝາະ​ສົມ​ຢ່າງ​ແນ່ນອນ ແຕ່​ຖ້າ​ເປັນ​ລະບົບ​ທົ່ວ​ໆ​ໄປ​ ຄວນ​ຈະ​ນຳ NoSQL ມາ​ໃຊ້​ງານ ​ຫຼື ​ບໍ່

    ຄຳ​ຕອບ​ຄື ຂຶ້ນ​ຢູ່​ກັບ​ລັກສະນະ​ງານ​ວ່າ ເຮົາ​ຈະ​ໃຊ້​ຄຸນສົມບັດ​ຫຍັງ​ຂອງ NoSQL ຖ້າ​ເຮົາ​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ບໍ່​ຕ້ອງ​ຢຶດ​ຕິດ​ກັບ​ໂຄງ​ສ້າງ (Dynamic Schema) ແລະ ​ຕ້ອງ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ລະບົບ​ທີ່​ວ່ອງໄວ (Integrated Caching) ຂໍ້​ມູນ​ອາດຈະ​ຢັງ​ບໍ່​ຫຼາຍ​ເທົ່າ​ໃດ ເຈົ້າ​ອາດຈະ​ໃຊ້ NoSQL ໄດ້​ຢ່າງ​ແນ່ນອນ

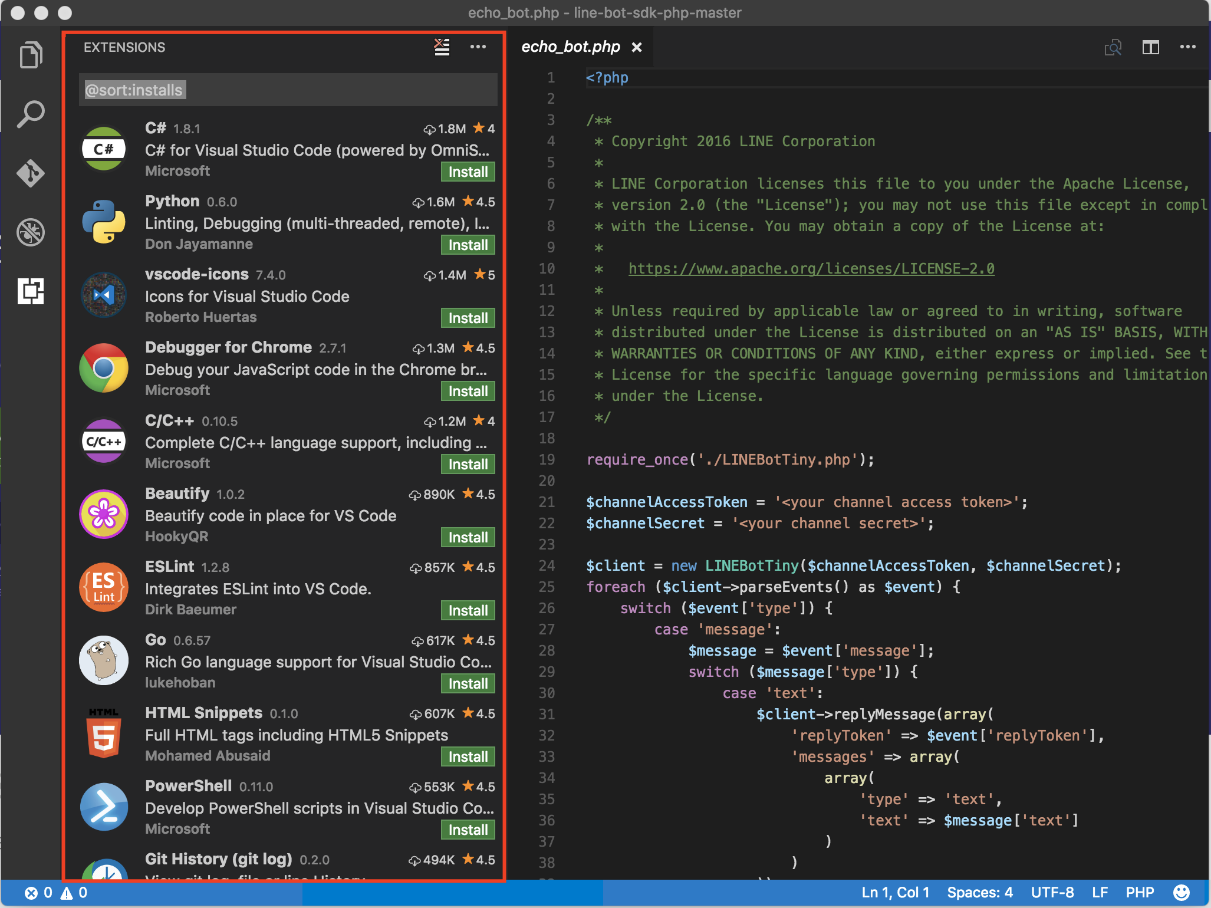
ແຕ່​ຖ້າ​ບໍ່​ຕ້ອງ​ການ​ໃຊ້​ງານ (Dynamic Schema) ບໍ່​ຕ້ອງ​ການ​ເຂົ້າ​ໃຊ້​ງານ​ທີ່​ວ່ອງໄວ (Integrated Caching) ເພາະ​ໃຊ້ Relational Database ກໍ​ເຮັດ​ໄດ້​ດີ​ຢູ່​ແລ້ວ Database Server ກັບ Web Server ກໍ​ຢູ່​ທີ່ Server ດຽວ​ກັນ ຂໍ້​ມູນ​ບໍ່​ຫລາຍ​ນັ້ນ ບໍ່​ຕ້ອງ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ທີ່​ເພີ່ມ​ຂະຫຍາຍ​ຂຶ້ນ​ທຸກ​ມື້​ໆ ຜູ້​ເຂົ້າ​ໃຊ້​ງານ​ກໍ​ບໍ່​ໄດ້​ຫລາຍ ເບິ່ງ​ແລ້ວ​ວ່າ​ລະບົບ​​ບໍ່​ມີ​ແນວ​ໂນ້ມ​ຈະ​ຕ້ອງ​ຂະຫຍາຍ​ລະບົບ​ໃນ​ອະນາຄົດ​ອັນ​ໃກ້ ເຈົ້າ​ສາມາດ​ໃຊ້​ງານ Relational Database ໄດ້​ດີ​ຢູ່​ແລ້ວຢ່າງ​ບໍ່​ມີ​ບັນຫາ.

* + - 1. ລະບົບຈັດການຖານຂໍ້ມູນ (Database Management Systems: DBMS)
    1. ພາສາທີ່ໃຊ້ຂຽນໂປຣແກຣມ
       1. ໂປຣແກຣມ Visual Studio Code

[Visual Studio Code](https://code.visualstudio.com/) ຫຼື ທີ່​ຫຼາຍ​ຄົນ​ນິຍົມ​ຮຽກ​ຫຍໍ້​ໆ​ເປັນ “[vs code](https://code.visualstudio.com/)” ຂໍ​ບອກ​​ກ່ອນ​ວ່າ​ Editor ໂຕ​ນີ້​ມັນ​ອອກ​ມາ​ຕັ້ງ​ແຕ່ 29 ເມ​ສາ ປີ 2015 ແລ້ວ​ ພັດທະນາ​ຂຶ້ນ​ໂດຍ​ບໍລິສັດ​ຍັກ​ໃຫຍ່​ນາມ ​ໄມ​ໂຄຣ​ຊອບ(Microsoft) ເປັນ​ທັງ​ໂຕ​ແກ້​ໄຂ​ ແລະ ​ປັບ​ແຕ່ງ​ໂຄດ (code optimized editor) ທີ່​ຕັດ​ຄວາມ​ສາ ມາດ​ມາ​ຈາກ Visual Studio ລຸ້ນ​ປົກກະຕິ (ພວກ GUI designer) ອອກ​ໄປ ເຫຼືອ​ແຕ່ໂຕ editor ຢ່າງ​ດຽວ ສາມາດ​ເຮັດວຽກ​ໄດ້​ຂ້າມ​ແພລດ​ຟອມທັງ​ໝົດວິນ Windows, Mac ແລະ​ Linux​ ຊັບ​ພອດພາ ສາ​ຫຼາຍ​ຮ້ອຍ​ພາສາ​ອີກ​ດ້ວຍ ​ຊຶ່ງ​ທາງ​ໄມ​ໂຄຊອບເອງ​ນັ້ນ​ໄດ້​ເປີດ​ໃຫ້​ໃຊ້​ຟີ​ອີກ​ດ້ວຍ.

ຄວາມ​ສາມາດ​ຂອງ​“vs code” ນັ້ນ​ຈະ​ມີ​ຄວາມ​ສາມາດ​ໃນ​ການ​ເປີດໄດ້​ຄືກັບ editor ໂຕ​ອື່ນ​ໆ ເຊັ່ນ sublime, Atom, Notepad++​ ນັ້ນ​ເອງ ທັງໝົດ​ເຖິງ​ຄວາມ​ສາມາດ​ໃນ​ການ​ຕິດ​ຕັ້ງ​ເຄື່ອງ​ມື​ເສີມ (Extension) ໂດຍ​ຮັບ ຮອງ​ໄດ້​ວ່າ​ມີ​ຊັບພອດຢ່າງແນ່ນອນເພາະ​ວ່າ​ມັນ​ຖືກ​ພັດທະນາ​ມາ​ໃຫ້​ຕອບ​ໂຈ​ດນັກ​ພັດທະນາ​ຫຼາຍ​ທີ່ສຸດ ແມ່ນການ​ດີ​ຊາຍໜ້າ​ຕາ ໃຫ້ເປັນ​ຮູບ​ແບບ​ທີ່​ເຂົ້າໃຈ ​ແລະ ​ໃຊ້​ງານ​ໄດ້​ງ່າຍ ບໍ່​ຕ້ອງ​ສຶກ​ສາ​ຫຍັງເພີ່ມເຕີມ​ກໍ​ໃຊ້​ງານ​ໄດ້ເລີຍ ສິ່ງ​ທີ່​ເຮັດໃຫ້​ມັນ​ໂດນ​ເດັ່ນ​ກວ່າ​ໂຕ​ອື່ນ​ໆ ຄື​ການ​ທີ່​ອອກ​ແບບ​ໃຫ້​ການ​ຄົ້ນ​ຫາ​ສິ່ງ​ຕ່າງ​ໆ ເຮັດ​ອອກ​ມາ​ໃຫ້​ໃຊ້​ງານ​ໄດ້​ງ່າຍ​ແລະ​ເບິ່ງ​ງ່າຍ​ກວ່າ​ໂຕອື່ນ​ໆ ທັງໝົດເຖິງ​ການ​ທີ່​ສ້າງ​ໃຫ້​ສາມາດ​ເຊື່ອມ​ຕໍ່​ກັບ Git ໄດ້​ຢ່າງ​ວ່ອງໄວ​ ແລະ ​ງ່າຍ​ດາຍ​ ມີ​ຟັງ​ຊັນໃນ​ການ commit, push & pull ຢູ່ໃນ​ໂຕ ຫຼື ​ຈະ​ເບິ່ງ change ຂອງ​ຟາຍ​ທີ່ເກິດຂື້ນ​ກໍ​ໄດ້​ແບບ​ງ່າຍດາຍບໍ່ຕ້ອງສຶກສາຫຍັງເພີ່ມແຕ່ກໍ່ໃຊ້ງານໄດ້ເລີຍ ([ມາ​ຮູ້​ຈັກ “VS Code” ອາ​ວຸດ​ຄູ່​ມື​ສາຍ​ພັດທະນາ​ຂອງ​ໃຜ​ຫຼາຍ​ໆ​ຄົນ](https://medium.com/@vortj/%E0%B8%A1%E0%B8%B2%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-vs-code-%E0%B8%AD%E0%B8%B2%E0%B8%A7%E0%B8%B8%E0%B8%98%E0%B8%84%E0%B8%B9%E0%B9%88%E0%B8%A1%E0%B8%B7%E0%B8%AD%E0%B8%82%E0%B8%AD%E0%B8%87%E0%B8%AA%E0%B8%B2%E0%B8%A2%E0%B8%9E%E0%B8%B1%E0%B8%92%E0%B8%99%E0%B8%B2%E0%B8%82%E0%B8%AD%E0%B8%87%E0%B9%83%E0%B8%84%E0%B8%A3%E0%B8%AB%E0%B8%A5%E0%B8%B2%E0%B8%A2%E0%B9%86%E0%B8%84%E0%B8%99-c6f476c0f60c)

April 3, 2017​).



ຮູບທີ 9 ໂປຣແກຣມ VSCode

* + - 1. ພາສາ JavaScript

ພາສາ JavaScript ຫລື​ຫຍໍ້ JS ເປັນ​ພາສາ​ຂຽນ​ໂປຣ​ແກຣມ​ທີ່​ຖືກ​ພັດທະນາ​ແລະ​ປະຕິບັດ​ຕາມ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ​ຂອງ ECMAScript; ພາສາ JavaScript ນັ້ນ​ເປັນ​ພາສາ​ລະ​ດັບ​ສູງ ຄອມ​ພາຍ​ໃນ​ຂະນະ​ທີ່​ໂປຣ​ແກຣມ​ຣັນ(JIT) ແລະ​ ເປັນ​ພາສາ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ຫຼາຍ​ຂະ​ບວນ​ເຊັ່ນ: ການ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ຂັ້ນ​ຕອນ, ການ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ວັດ​ຖຸ, ພາສາ JavaScript ມີ​ໄວ​ຍະກອນທີ່​ຄື​ກັບ​ພາສາ C ໃຊ້​ວົງ​ເລັບ​ເພື່ອ​ກຳນົດ​ບ່ອກ​ຂອງ​ຄຳ​ສັ່ງ ນອກ​ຈາກ​ນີ້ JavaScript ຢັງ​ເປັນ​ພາສາ​ທີ່​ມີ​ປະ​ເພດ​ຂໍ້​ມູນ​ແບບ​ໄດ​ນາ​ມິກ (Dynamic)​ ເປັນ​ພາສາ​ແບບ Prototype-based ແລະ First-class function.

ພາສາ JavaScript ນັ້ນ​ຖື​ວ່າ​ເປັນ​ເທັກ​ໂນ​ໂລ​ຊີ​ຫຼັກ​ຂອງ​ການ​ພັດທະນາ​ເວັບ​ໄຊ​ (World Wide Web) ມັນ​ເຮັດໃຫ້​ໜ້າ​ເວັບ​ສາມາດ​ຕອບ​ໂຕ້​ກັບ​ຜູ້​ໃຊ້​ໄດ້​ໂດຍ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ລີເຟດໜ້າ​ໃໝ່ (Dynamic website) ເວັບ​ໄຊ​ຈຳນວນ​ຫຼາຍ​ໃຊ້​ພາສາ JavaScript ສຳລັບ​ຄວບ​ຄຸມ​ການ​ເຮັດ​ວຽກງານ​ທີ່​ດ້ານ Client-side ນັ້ນ​ເຮັດໃຫ້​ເວັບ​ເບ​ຣາ​ວ​ເຊີຕ່າງ​ໆ ມີ JavaScript engine ທີ່​ໃຊ້​ສຳລັບ​ປະ​ມວນ​ຜົນສະຄິບ​ຂອງ​ພາສາ JavaScript ທີ່​ຣັນ​ເທິງ​ເວັບ​ບາວ​ເຊີເນື່ອງ​ຈາກ​ພາສາ JavaScript ເປັນ​ພາສາ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບ​ຫຼາຍ​ຮູບແບບ ເຮັດໃຫ້​ມັນ​ຮອງ​ຮັບ​ການ​ຂຽນ​ໂປຣ​ແກຣມ​ທັງ​ແບບ Event-driven, Functional ແລະ ​ແບບ​ລຳ​ດັບ​ຂັ້ນ​ຕອນ ມັນ​ມີ​ Library (APIs) ສຳລັບ​ເຮັດວຽກ​ກັບ​ຂໍ້ຄວາມ, ວັນທີ, Regular expression ແລະ ​ໂຄງ​ສ້າງ​ຂໍ້​ມູນ​ພື້ນ​ຖານ​ຢ່າງ Array ແລະ Map ຫຼື ​ແມ່ນ​ທັງ Document Object Model (DOM) ຊຶ່ງ​ເປັນ API ທີ່​ໂດຍ​ທັ່ວ​ໄປ​ແລ້ວ​ສາມາດ​ໄດ້​ເທິງ​ເວັບບາວເຊີ.

ຢ່າງໃດ​ກໍ​ຕາມ ຕົວ​ຂອງ​ພາສາ JavaScript ເອງ​ບໍ່​ໄດ້​ມີ​ຟັງ​ຊັນ​ສຳລັບອິນພຸດ/ເອົາພຸດ(I/O) ທີ່​ມາ​ກັບ​ພາສາເຊັ່ນ ຟັງ​​ຊັນ​ກ່ຽວກັບ Network ວຽກກ່ຽວ​ກັບ​ໄຟ​ລ ຫຼື Library​ ກ່ຽວກັບ​ກຣາ​ຟິກ ໂດຍ​ທົ່ວ​ໄປ​ແລ້ວ​ສິ່ງເຫຼົ່ານີ້​ຈະ​ຖືກ​ໃຫ້​ມາ​ໂດຍ Host environment (ສະພາບ​ແວດ​ລ້ອມ​ທີ່​ໃຊ້​ຣັນ​ພາສາ JavaScript) ເຊັ່ນ ເວັບ​​ເວັບບາວເຊີ ຫຼື Node.js ຊຶ່ງ​ຈະ​ແຕກ​ຕ່າງ​ກັນ​ອອກ​ໄປ ​ຕົວ​ຢ່າງ​ເຊັ່ນ ການ​ຮັບ​ຄ່າ​ໃນ​ເວັບ​​ເວັບບາວເຊີ​ຈະ​ຜ່ານ​ຟັງ​ຊັນ prompt ຊຶ່ງ​ເປັນ​ສ່ວນຫນຶ່ງ​ຂອງ Browser Object Model (BOM) ຫລື​ຮັບ​ຄ່າ​ຈາກ HTML ຟອມຊຶ່ງ​ເປັນ​ສ່ວນຫນຶ່ງ​ຂອງ Document Object Model (DOM) ໃນຂະນະ​ທີ່​ເທິງ Node.js ເຮົາ​ສາມາດ​ຮັບ​ຄ່າ​ໄດ້​ຈາກ Input/Output Stream ຂອງ Command line ເຖິງ​ແມ່ນ​ວ່າ​ມັນ​ຈະ​ມີ​ຄວາມ​ຄ້າຍ​ຄືກັນ​ລະຫວ່າງ​ພາສາ Java ແລະ JavaScript ເຊັ່ນ: ຊື່​ຂອງ​ພາສາ ໄວ​ຍະກອນຫລື​ Library​ ​ມາດຕະຖານ​ຕ່າງ​ໆ ຢ່າງໃດ​ກໍ​ຕາມ ທັງ​ສອງ​ພາສາ​ແຕກ​ຕ່າງ​ກັນ​ຢ່າງ​ສິ້ນ​ເຊີງ​ໃນ​ແງ່​ຂອງ​ການ​ອອກ​ແບບ ພາສາ Java ເປັນ​ພາສາ​ທີ່​ມີ​ປະ​ເພດ​ຂໍ້​ມູນ​ແບບ​ຄົງ​ທີ່ (Static-typing) ໃນ​ຂະນະ​ທີ່​ພາສາ JavaScript ມີ​ປະ​ເພດ​ຂໍ້​ມູນ​ແບບ​ໄດ​ນາ​ມິກ​ (Dynamic-typing) ພາສາ Java ຖືກ​ຄອມ​ພາຍເປັນ Byte-code ກ່ອນທີ່ຈະ​ລັນ ໃນ​ຂະນະ​ທີ່​ພາສາ JavaScript ຈະ​ຄອມ​ພາຍໃນ​ຕອນ​ທີ່​ໂປຣ​ແກຣມ​ລັນພາສາ Java ເປັນ​ພາສາ​ແບບ Class-based ໃນ​ຂະນະ​ທີ່​ພາສາ JavaScript ເປັນ​ພາສາ​ແບບ Prototypebased



ຮູບທີ 10 ພາສາ JavaScript

ກ. ປະຫວັດ​ຄວາມ​ເປັນ​ມາ​ຂອງ​ພາສາ JavaScript

ພາສາ JavaScript ຖືກ​ອອກ​ແບບ​ແລະ​ສ້າງ​ໂດຍ Brendan Eich ສຳລັບ​ເປັນ​ພາສາ​ສະຄິບ​ທີ່​ເຮັດວຽກ​ເທິງ​ເວັບ​ເບ​ຣາ​ວ​ເຊີ Navigator ທີ່​ເປັນ​ຜະລິດຕະພັນ​ຂອງ​ບໍລິສັດ Netscape ເພື່ອ​ເຮັດໃຫ້​ໜ້າ​ເວັບ​ທີ່​ໃນ​ຕອນ​ທຳອິດ​ນັ້ນ​ເປັນ​ແບບ Static ສາມາດ​ຕອບ​ໂຕ້​ກັບ​ຜູ້​ໃຊ້​ໄດ້​ໂດຍ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ຣີ​ເຟ​ດ​ໜ້າ​ໃໝ່ (Dynamic) ເຊັ່ນ ການ​ສົ່ງ​ຂໍ້​ມູນ​ເບື້ອງ​ຫຼັງ​ໄປ​ຢັງ Server ແລະ​ລໍຖ້າ​ຮັບ​ຜົນ​ຕອບ​ກັບ​ມາ​ດ້ວຍ AJAX; ພາສາ JavaScript ໄດ້​ຖືກ​ເປີດ​ຕົວ​ ແລະ ​ເປັນ​ສ່ວນຫນຶ່ງ​ຂອງ​ເວັບ​ບ​າ​ວ​ເຊີ Navigator ໃນ​ເດືອນ​ກັນຍາ 1995 ໂດຍ​ໃຊ້​ຊື່​ວ່າ LiveScript ແລະ ​ໄດ້​ປ່ຽນ​ເປັນ JavaScript ໃນ​ອີກ​ສາມ​ເດືອນ​ຕໍ່​ມາ.

ໃນ​ເດືອນ​ພ​ະຈິກ 1996 Netscape ໄດ້​ສົ່ງ​ພາສາ JavaScript ໄປ​ຢັງ ECMA International ເພື່ອ​ເປັນ​ຈຸດ​ເລີ່ມ​ຕົ້ນ​ສຳລັບ​ກຳນົດ​ມາດຕະຖານ​ໃຫ້​ທຸກ​ເວັບ​ບາວເຊີປະຕິບັດຕາມ​ມາດຕະຖານ​ດັ່ງ​ກ່າວ ເພື່ອ​ໃຫ້​ການ​ພັດທະນາ JavaScript engine ເປັນ​ໄປ​ໃນ​ທິດ​ທາງ​ດຽວ​ກັນ ນັ້ນ​ໃຫ້​ເກີດ​ການ​ເປີດ​ຕົວ​ຢ່າງ​ເປັນ​ທາງ​ການ​ສຳລັບ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ ECMAScript ໃນ​ເດືອນ​ມິ​ຖຸ​ນາ 1997 ໃນ​ຊ່ວງ​ເວລາ​ຫຼັງ​ຈາກ​ນີ້ ບໍລິສັດ​ຕ່າງ​ໆ ທີ່​ພັດທະນາ​ເວັບ​ບາ​ວ​ເຊີຕ່າງ​ກໍ​ຍັງ​ພັດທະນາ JavaScript engine ບໍ່​ເປັນ​ໄປ​ໃນ​ທິດ​ທາງ​ດຽວ​ກັນ​ເທົ່າໃດ ນັ້ນ​ເຮັດໃຫ້​ນັກ​ພັດທະນາ​ເວັບ​ຕ້ອງ​ຂຽນ​ໂຄດຫຼາຍ​ເວີຊັນ​ເພື່ອ​ໃຫ້​ເຮັດວຽກ​ໄດ້​ໃນ​ທຸກ​ເວັບ​ບ​າ​ວ​ເຊີຈົນກະທັ້ງ​ໃນ​ເ​ດືອນກໍລະກົດ 2008 ໄດ້​ມີ​ການ​ຈັດການ​ປະ​ຊຸມ​ຂຶ້ນ​ທີ່ Oslo ຈາກ​ອົງ​ກ​ອນ​ແລະ​ຝ່າຍ​ຕ່າງ​ໆ ທີ່​ພັດທະນາ JavaScript engine ເຮັດໃຫ້​ເກີດ​ຂໍ້​ຕົກລົງ​ຂຶ້ນ​ໃນ​ຕົ້ນ​ປີ 2009 ເພື່ອ​ລວບລວມ​ງານ​ທີ່​ກ່ຽວ​ຂ້ອງ​ທັງ​ໝົດ​ຂອງ​ພາສາ JavaScript ແລະ ​ຊຸກຍູ້​ພາສາ​ໃຫ້​ຍ່າງ​ໄປ​ຂ້າງ​ໜ້າ ນັ້ນ​ເຮັດໃຫ້​ເກີດ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ ECMAScript ເວີຊັນ​ທີ່ 5 (ES5) ອອກ​ມາ​ໃນ​ເດືອນ​ທັນວາ 2009 ແລະ ​ກ່ອນ​ໜ້າ​ນີ້​ໃນ​ປີ 2008 Google ໄດ້​ເປີດ​ຕົວ​ເວັບ​ບ​າ​ວ​ເຊີ Chrome ທີ່​ມາ​ພ້ອມ​ກັບ V8 JavaScript engine ທີ່​ມີ​ແນວ​ຄິດ​ໃນ​ການ​ພັດທະນາ​ແບບ​ຄອມ​ພາຍ​ໃນ​ຕອນ​ທີ່​ໂປຣ​ແກຣມ​ລັນ (Just-in-time compilation: JIT) ຊຶ່ງ​ມັນ​ເຮັດວຽກ​ໄດ້​ໄວ​ກວ່າ​ຫຼາຍ ເຮັດໃຫ້​ຜູ້​ພັດທະນາ​ເວບບ​າ​ວ​ເຊີ​ອື່ນ​ໆ ຕ້ອງ​ປັບ​ປຸງ JavaScript engine ຂອງ​ພວກ​ເຂົາ​ໃຫ້​ເຮັດວຽກ​ໃນ​ຮູບ​ແບບ JIT ຫຼັງ​ຈາກ​ທີ່​ພັດທະນາ​ຕໍ່​ເນື່ອງ​ມາ​ອີກ​ຫຼາຍ​ປີ ໃນ​ປີ 2015 ໄດ້​ມີ​ການ​ເພີ່ມ​ຄຸນສົມບັດ​ໃໝ່​ໆ ທີ່​ຫຼາກ​ຫຼາຍ​ເຂົ້າ​ມາ ຊຶ່ງ​ຖື​ວ່າ​ເປັນ​ການ​ປ່ຽນ​ແປງ​ເທື່ອ​ສຳຄັນ ແລະ​ເຮັດໃຫ້​ເກີດ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ ECMAScript 2015 ຫຼື ​ເວີຊັນທີ່ 6 (ES6) ຈົນກະທັ້ງ​ໃນ​ປີ 2015 ຕອນ​ນີ້​ເບິ່ງ​ຄື​ວ່າ​ພາສາ JavaScript ຈະ​ພັດທະນາ​ມາ​ຈົນ​ເຖິງ​ທີ່ສຸດ​ແລ້ວ ເຮັດໃຫ້​ລະຫວ່າງ​ປີ 2016 - 2019 ເວີຊັນ​ໃໝ່​ຂອງ ECMAScript ທີ່​ຖືກ​ເຜີຍແຜ່​ອອກ​ມາ​ໃນ​ແຕ່​ລະ​ປີ​ມີ​ການ​ປ່ຽນ​ແປງ​ແລະ​ເພີ່ມ​ຄຸນສົມບັດ​ພຽງ​ນ້ອຍໆ​ເທົ່າ​ນັ້ນ.

ຂ. ຄຸນສົມບັດ​ຂອງ​ພາສາ JavaScript

ECMAScript 2015 (ES6) ເປັນ​ພາສາ JavaScript ທີ່​ຖື​ວ່າ​ພັດທະນາ​ມາ​ຈົນ​ເຖິງ​ຈຸດ​ສູງ​ສຸດ​ແລ້ວ​ກໍ​ວ່າ​ໄດ້ ມັນ​ຖືກ​ເຜີຍແຜ່​ໃນ​ເດືອນ​ມິ​ຖຸ​ນາ 2015 ຊຶ່ງ​ໃນ​ເວີຊັນ​ນີ້ ໄດ້​ເພີ່ມ​ໄວ​ຍະກອນໃໝ່​ຂອງ​ພາສາ​ຫຼວງ​ຫລາຍເຊັ່ນ ການສ້າງ​ຄາດ​ດ້ວຍ​ຄຳ​ສັ່ງ class ການສ້າງ​ໂມ​ດູນ​ແລະ​ໃຊ້​ງານ​ມັນ​ດ້ວຍ​ຄຳ​ສັ່ງ import ແລະ export ແລະ​ ຄຳ​ສັ່ງ​ສຳລັບ​ປະ​ກາດ​ຕົວ​ປ່ຽນ let ແລະ ​ປະ​ກາດ​ຄ່າ​ຄົງທີ const ຊຶ່ງ​ເຮັດໃຫ້​ຕົວ​ປ່ຽນສາມາດ​ມີ​ຂອບ​ເຂດ​ໃນ​ບ​ລັອກ​ທີ່​ມັນ​ຖືກ​ສ້າງ​ຂຶ້ນ​ໄດ້ ແລະ​ສິ່ງ​ອື່ນ​ໆ ທີ່​ຖືກ​ເພີ່ມ​ເຂົ້າ​ມາ​ເປັນ​ຈຳນວນ​ຫລາຍ ເຊັ່ນ Map, Set, WeakMap, Promise, Reflection, Proxies, Template string ແລະ​ອື່ນ​ໆ.

ໃນ​ເດືອນ​ມິ​ຖຸ​ນາ 2016 ໄດ້​ມີ​ການ​ເປີດ​ຕັວ​ເວີຊັນ 7 ຫລື ECMAScript 2016 (ES7) ໄດ້​ມີ​ການ​ເພີ່ມ​ຕົວ​ດຳ​ເນີນ​ການ​ຍົກ​ກຳ​ລັງ \*\* (ທີ່​ກ່ອນ​ໜ້າ​ນີ້​ເຮົາ​ຈະ​ໃຊ້​ຜ່ານ​ຟັງຊັນ Math.pow) ຄຳ​ສັ່ງawait async ສຳລັບ​ການ​ຂຽນ​ໂປຣ​ແກຣມ​ທີ່​ເຮັດວຽກ​ບໍ່​ພ້ອມ​ກັນ ແລະ ​ຟັງຊັນ includes ຂອງ​ອາ​ເລແລະ ​ໃນ​ປະຈຸບັນ ພາສາ JavaScript ຖືກ​ພັດທະນາ​ມາ​ຈົນ​ເຖິງ ECMAScript 2020 (ES11) ຊຶ່ງ​ມີ​ການ​ປ່ຽນ​ແປງ​ທີ່​ເພີ່ມ​ຂຶ້ນ​ບໍ່​ເທົ່າໃດ​ຫຼັງ​ຈາກ ES7

ຄ. JavaScript engine ແມ່ນຫຍັງ?

JavaScript engine ຄື​ໂປຣ​ແກຣມ​ຄອມພິວເຕີ​ທີ່ໃຊ້​ສຳລັບ​ປະ​ມວນ​ຜົນ​ໂຄດຂອງ​ພາສາ JavaScript ຊຶ່ງ JavaScript engine ໃນ​ຊ່ວງ​ເລີ່ມ​ຕົ້ນ​ເປັນພຽງ​ແຄ່​ຕົວປ່ຽນພາສາ (Interpreter) ເທົ່າ​ນັ້ນ ແຕ່​ໃນ​ປະຈຸບັນ​ໄດ້​ມີ​ການ​ພັດທະນາ​ມາ​ໃຫ້​ຢູ່ໃນ​ຮູບ​ແບບ​ຂອງ​ຄອມ​ພາຍເລິ​ທີ່​ມີ​ການ​ຄອມ​ພາຍໃນ​ຕອນ​ທີ່​ໂປຣ​ແກຣມ​ລັນ (Just-in-time compilation: JIT) ເພື່ອ​ເພີ່ມ​ປະ​ສິດ​ທິ​ພາບ​ການ​ເຮັດ​ວຽກງານ​ຂອງ​ໂປຣ​ແກຣມ ໂດຍ​ທົ່ວ​ໄປ​ແລ້ວ JavaScript engine ຈະ​ຖືກ​ພັດທະນາ​ໂດຍ​ຜູ້​ພັດທະນາ​ເວັບບາວເຊີ້ທີ່​ປະຕິບັດ​ຕາມ​ຂໍ້​ກຳນົດ​ມາດຕະຖານ​ຂອງ ECMAScript (ແນະນຳພາສາ JavaScript <http://marcuscode.com/lang/javascript/introducing-to-javascript> 28 July 2020​).

* + - 1. Reactjs

React ເປັນເທັກໂນໂລຢີໜຶ່ງທີ່ມາແຮງຫຼາຍໂດຍສ້າງມາຈາກພື້ນຖານແນວຄວາມຄິດແບບ MVC (Model View Controller) ຊຶ່ງໝາຍຄວາມວ່າ React ມີຫນ້າທີ່ຈັດການກັບ Model ຫຼືView ແຕ່ສ່ວນໃຫຍ່ຈະເປັນ View ກ່ອນໜ້ານັ້ນເວລາຈະຂຽນໜ້າເວັບ ເຮົາກໍຈະເຮັດຜ່ານ HTML ມີການໃຊ້ CSS ໃນການປັບປ່ຽນໜ້າຕາຂອງ UI ແຕ່​ໃນ React ຖ້າ​ຈະ​ສ້າງ​ໜ້າ​ເວັບ​ຂຶ້ນ​ມາ​ໄດ້​ນັ້ນ ເຮົາ​ຈະ​ໃຊ້​ເປັນ [Component](https://www.mindphp.com/%E0%B8%9A%E0%B8%97%E0%B9%80%E0%B8%A3%E0%B8%B5%E0%B8%A2%E0%B8%99%E0%B8%AD%E0%B8%AD%E0%B8%99%E0%B9%84%E0%B8%A5%E0%B8%99%E0%B9%8C/tutorial-reactjs/4864-component-and-props.html) ປຽບໄດ້​ວ່າ Component ເປັນ Block ສ່ວນ​ຍ່ອຍ​ຂອງ​ເວັບ​ເຮົາ​ທີ່​ຈະ​ສ້າງ​ອອກ​ມາ. (ມາທຳຄວາມຮູ້ຈັກ React https://www.mindphp.com/บทเรียนออนไลน์/tutorial-reactjs/4883-react.html 13 May 2017).

ສະຫລຸບ​ງ່າຍໆ​ດັ່ງ​ນີ້

* React ເປັນ Javascript Libraly ສ້າງ​ແລະ​ພັດທະນາ​ຂຶ້ນ​ໂດຍ Facebook ຂຽນ​ໄດ້​ແຕ່ UI ເທົ່າ​ນັ້ນ ແລະ​ເປີດ​ໃຫ້​ໃຊ້​ຟ​ຣີ
* React ໃຊ້​ໂຄດ HTML , CSS ແລະ Javascript
* React ມີ 3 ຄອນ​ເຊບ​ທີ່​ເຮົາ​ຕ້ອງ​ຮຽນ​ຮູ້ ຄື Component State ແລະ Props
* ຂໍ້​ດີ​ແລະ​ຂໍ້​ເສຍ​ຂອງ React

ກ. ຂໍ້​ດີ

* Component ເຂົ້າໃຈ​ງ່າຍ ເຮົາ​ສາມາດ​ຮຽນ​ຮູ້​ໄດ້​ດ້ວຍ​ຕົວ​ເອງ
* Tool ຫຼາຍ ພຽງReact ຢ່າງ​ດຽວ​ກໍ່ສາມາດຂຽນເວັບ​ໄດ້​ທັງ​ເວັບ​ແລ້ວ ໂດຍ​ບໍ່​ຕ້ອງ​ຫາ Tool ເພີ່ມເຕິມ ແລະ​ຍັງມີ Tool ພັດທະນາ​ອອກ​ມາ​ຢູ່​ເລື້ອຍໆສາມາດ​ເຮັດ App ໄດ້ React ມີ​ເຄື່ອງ​ມື​ໜຶ່ງ​ທີ່​ຊື່​ວ່າ React Native ເປັນ​ການ​ຂຽນ JavaScript ແລ້ວ​ແປງ​ເປັນ App ແບບ Native ໄດ້​ທັງ​ເທິງ Android ແລະ iOS

ຂ. ຂໍ້​ເສຍ

* ຕ້ອງ​ມີ​ພື້ນ​ຖານ​ໃນ Javascript ໃນ​ລະ​ດັບ​ໜຶ່ງ ຄົົນທີ່​ສົນ​ໃຈ​ຮຽນ​ກໍ​ຈະ​ລຳ​ບາກ​ນ້ອຍ​ນຶງ​ຖ້າ​ບໍ່​ໄດ້​ມີ​ພື້ນ​ຖານ Javascript ອາດຈະ​ຕ້ອງ​ໃຊ້​ເວລານ້ອຍໜື່ງ
* Documentation ອ່ານ​ຍາກ React ມີ Documentation ທີ່​ຍັງ​ບໍ່​​ຄ່ອຍ​ດີ ໃນ​ອະນາຄົດ​ອາດຈະ​ມີ​ການ​ພັດທະນາ​ອີກ
  + - 1. Nodejs

Node.js ເປັນ open-source ແລະ cross-platform JavaScript runtime environment ທີ່​ກຳ​ລັງ​ໄດ້ຮັບ​ຄວາມ​ນິຍົມ​ສູງ ໂດຍ​ທົ່ວ​ໄປ​ເຮົາ​ຈະ​ໃຊ້ JavaScript ໃນ​ຝັ່ງ client ແຕ່ Node.js ເຮັດໃຫ້​ເຮົາ​ໃຊ້ JavaScript ໃນ​ຝັ່ງ Server ໄດ້​ດ້ວຍ ຊຶ່ງ Node.js ສາມມາດ run​ ໄດ້​ເທິງ platform ທີ່​ຫຼາກ​ຫຼາຍ​ທັງ Windows, Linux, Unix, Mac OS X ແລະ ອື່ນໆ.

ຮູບທີ 12 Nodejs

ຈຸດ​ເດັ່ນ​ທີ່ສຸດ​ຂອງ Node.js ຄື​ມັນ​ເຮັດວຽກ​ແບບ asynchronous ຜູ້​ອ່ານ​ອາດຈະ​ສົງ​ໃສ​ວ່າ​ແລ້ວ​ມັນ​ດີ​ແນວໃດ? ລອງ​ມາ​ເບິ່ງ​ຕົວ​ຢ່າງ​ການ​ຈັດການ​ໄຟ​ລ​ເມື່ອ​ມີ​ການ​ຮ້ອງ​ຂໍ​ຈາກ client ມາ​ທີ່ server ຂອງ PHP ກັບ Node.js ທຽບ​ກັນ​ເບິ່ງ​ເພື່ອ​ໃຫ້​ເຂົ້າໃຈ​ຫລາຍ​ຂຶ້ນ​

* ກໍລະນີ PHP ເມື່ອ​ມີ​ການ​ຮ້ອງ​ຂໍ​ເຂົ້າ​ມາ​ມັນ​ຈະ​ເຮັດ​ດັ່ງ​ນີ້:

1. ສົ່ງ task ໄປ​ທີ່​ລະບົບ​ໄຟ​ຂອງ​ຄອມພິວເຕີ
2. ລໍຖ້າຈົນກະທັ້ງ​ລະບົບ​ໄຟ​ລ​ເປີດ​ແລະ​ອ່ານ​ໄຟ​ລ​ສຳເລັດ
3. ສົ່ງ​ເນື້ອ​ຫາ​ຂອງ​ໄຟ​​ກັບ​ມາ​ໃຫ້ client
4. ພ້ອມ​ສຳລັບ​ຮັບ​ຄຳ​ຮ້ອງ​ຂໍ​ຖັດ​ໄປ

* ກໍລະນີ Node.js ເມື່ອ​ມີ​ການ​ຮ້ອງ​ຂໍ​ເຂົ້າ​ມາ​ມັນ​ຈະ​ເຮັດ​ດັ່ງ​ນີ້:

1. ສົ່ງ task ໄປ​ທີ່​ລະບົບ​ໄຟ​ຂອງ​ຄອມພິວເຕີ
2. ພ້ອມ​ສຳລັບ​ຮັບ​ຄຳ​ຮ້ອງ​ຂໍ​ຖັດ​ໄປ
3. ພໍ​ລະບົບ​ໄຟ​ລ​​ເປີດ​ແລະ​ອ່ານ​ໄຟ​ລ​ສຳເລັດ server ຈະ​ສົ່ງ​ເນື້ອ​ຫາ​ຂອງ​ໄຟ​ກັບ​ມາ​ໃຫ້ client

ຈາກ​ຕັວ​ຢ່າງ​ດ້ານ​ເທິງ​ຈະ​ເຫັນ​ວ່າ Node.js ຈະ​ຕັດ​ຂັ້ນ​ຕອນ​ການ​ລໍຖ້າຖີ້ມ​ແລ້ວ​ໄປ​ເຮັດ​ຄຳ​ຮ້ອງ​ຖັດ​ໄປ​ເລຍ ທີ່​ເປັນ​ແບບນີ້​ເພາະ Node.js ຈະ​ run ​ແບບ single-threaded ແລະ​ໃນ​ Libaryມາດຕະຖານ​ກໍ​ຈະ​ມີ​ເຊັດ​ຂອງ asynchronous I/O primitives ທີ່​ຊ່ວຍ​ປ້ອງ​ກັນ​ໂຄດ JavaScript ຈາກ​ການ blocking ເຮັດໃຫ້​ລະບົບ​ຄ່ອງແຄ້ວ ແລະ ​ມີ​ປະ​ສິດ​ທິ​ພາບ​ຫຼາຍ​ຂຶ້ນ

|  |  |
| --- | --- |
| Synchronous vs Asynchronous | |
| Synchronous ຄື​ການ​ run​ໂຄດ​ຕາມ​ລຳ​ດັບ​ທີ່​ເຮົາ​ຂຽນ​ໄວ້ ເຊັ່ນ  alert(1);  alert(2);  alert(3); ຜົນ​ລັບ​​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງ​ຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ສະແດງ​ຜົນ 2 ແລ້ວ​ສຸດ​ທ້າຍ​ຈຶ່ງ​ສະແດງ​ຜົນ 3 ຕາມ​ລຳ​ດັບ | Asynchronous ຄື​ການ​ຣັນໂຄ​ດ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເປັນ​ໄປ​ຕາມ​ລຳ​ດັບ​ທີ່​ເຮົາ​ຂຽນ​ໄວ້ ເຊັ່ນ:  alert(1);  setTimeout(() => alert(2), 0);  alert(3); ໃນ​ຕົວ​ຢ່າງ​ນີ້​ໂຄດ alert(2) ໃຊ້​ເວລາ​ດຳ​ເນີນ​ການ​ດົນ​ກວ່າ ເຮັດໃຫ້​ຜົນ​ລັບ​​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງ​ຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ສະແດງ​ຜົນ 3 ແລ້ວ​ສຸດ​ທ້າຍ​ຈຶ່ງ​ສະແດງ​ຜົນ 2 |
| Blocking vs Non-blocking | |
| Blocking ໝາຍ​ເຖິງ​ການ​ທີ່​ເຮົາ​ບໍ່​ສາມາດ​ດຳ​ເນີນ​ການ​ຕໍ່ໄປ​ໄດ້​ຈົນ​ກວ່າ​ຕົວ​ດຳ​ເນີນ​ການ​ທີ່​ກຳ​ລັງ​ລັນ​ຢູ່​ຈະ​ສຳເລັດ​ເສຍ​ກ່ອນ ເຊັ່ນ  alert(1);  var value = localStorage.getItem('foo');  alert(2); ຄຳ​ສັ່ງ localStorage ຈະ​ເປັນ​ຕົວ blocking ເຮັດໃຫ້​ບໍ່​ສາມາດ​ເຮັດ​ຄຳ​ສັ່ງ alert(2) ໄດ້​ຈົນ​ກວ່າ​ມັນ​ຈະ​ດຳ​ເນີນ​ການ​ສຳເລັດ ດັ່ງ​ນັ້ນ​ຜົນ​ລັບ​​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງ​ຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ລໍຖ້າຈົນກວ່າ​ຄຳ​ສັ່ງ localStorage ຈະ​ສຳເລັດ​ຈຶ່ງ​ສະແດງ​ຜົນ 2 | Non-blocking ໝາຍ​ເຖິງ​ການ​ໂຕທີ່​ດຳ​ເນີນ​ການ​ສາມາດ​ເຮັດ​ຄຳ​ສັ່ງ​ຖັດ​ໄປ​ໄດ້​ເລີຍ​ໂດຍ​ບໍ່​ຕ້ອງ​ລໍຖ້າ​ໃຫ້​ຄຳ​ສັ່ງ​ເດີມ​ເຮັດ​ສຳເລັດ​ກ່ອນເຊັ່ນ:  alert(1);  fetch('example.com').then(() => alert(2));  alert(3); ໃນ​ຕົວ​ຢ່າງ​ນີ້ ຄຳ​ສັ່ງ fetch ເປັນ non-blocking operation ດັ່ງ​ນັ້ນ​ຜົນ​ລັບ​ທີ່​ໄດ້​ຄື​ໜ້າ​ຈໍ​ຈະ​ສະແດງຜົນ 1 ຈາກ​ນັ້ນ​ຈຶ່ງ​ສະແດງ​ຜົນ 3 ແລ້ວ​ພໍ​ຄຳ​ສັ່ງ fetch ສຳເລັດ​ກໍ​ສະແດງ​ຜົນ 2 |

* ​ປະຫວັດ Node.js

ແຕ່​ເດີມ​ພາສາ JavaScript ຖືກ​ພັດທະນາ​ມາ​ເພື່ອ​ໃຊ້​ສຳລັບ Browser ທີ່​ຊື່ Netscape Navigator ໃນ​ປີ 1995 ​ໃນ​ຕອນ​ນັ້ນ Netscape ຕັ້ງ​ໃຈ​ຈະ​ຂາຍ Web Server ທີ່​ມີ environment ຊື່ Netscape LiveWire ຊຶ່ງ​ສາມາດ​ສ້າງ dynamic page ໂດຍ​ໃຊ້ JavaScript ທາງ​ຝັ່ງ server ດ້ວຍ ແຕ່​ໜ້າ​ເສຍ​ດາຍ​ທີ່ Netscape LiveWire ບໍ່​ປະ​ສົບ​ຄວາມ​ສຳ​ເລັດ ແລະ​ການ​ໃຊ້ JavaScript ທາງ​ຝັ່ງ server ກໍ​ບໍ່​ໄດ້ຮັບ​ຄວາມ​ນິຍົມ​ເລີຍ​ຈົນກະທັ້ງ Node.js ຖືກ​ກຳ​ເນີດ​ຂຶ້ນ​ມາ.

ສິ່ງ​ທີ່​ເຮັດໃຫ້ Node.js ເປັນ​ທີ່​ນິຍົມ​ຂຶ້ນ​ມາ​ຄື​ການ​ທີ່​ມັນ​ມາ​ໃນ​ຊ່ວງ​ເວລາ​ທີ່​ເໝາະ​ສົມ ເມື່ອ​ທຽບ​ກັບ JavaScript ທີ່​ເກີດ​ມາ​ຕັ້ງ​ແຕ່​ປີ 1995 ແລ້ວ Node.js ຫາກໍ່​ເກີດ​ມາ​ເມື່ອ​ປີ 2009 ເທົ່າ​ນັ້ນ ຕ້ອງ​ຂອບໃຈ "Web 2.0" applications (ເຊັ່ນ Flickr, Gmail ແລະ ອື່ນ ໆ) ທີ່​ສະແດງ​ໃຫ້​ໂລກ​ຮູ້​ວ່າ​ເວັບ​ສະໄໝ​ໃໝ່​ຄວນ​ໜ້າ​ຕາ​ເປັນ​ຢ່າງໃດ

ບໍ່ດົນ​ກ່ອນ​ທີ່ Node.js ຈະ​ເກີດ ນັກ​ພັດທະນາ​ເບື້ອງ​ຫຼັງ browser ຊື່​ດັ່ງ​ທັງ​ຫຼາຍ​ແຂ່ງ​ຂັນ​ກັນ​ເຮັດວຽກ​ຢ່າງ​ໜັກ​ເພື່ອຈະ​ໃຊ້ JavaScript ໃຫ້​ໄດ້​ດີ​ທີ່ສຸດ​ແລະ​ຫາ​ທາງ​ເຮັດໃຫ້ JavaScript ສາມາດ​ລັນ​ໄດ້​ໄວ​ຫລາຍ​ຂຶ້ນ ເພື່ອ​ໃຫ້​ຜູ້​ໃຊ້​ງານ​ໄດ້ຮັບ​ປະ​ສິດ​ທິ​ພາບ​ທີ່​ດີ​ທີ່ສຸດ ຊຶ່ງ​ຜົນ​ຈາກ​ການ​ແຂ່ງ​ຂັນ​ນີ້​ເຮັດໃຫ້​ເກີດ​ການ​ພັດທະນາ Chrome V8 (open-source JavaScript engine ຂອງ The Chromium Project) ຂຶ້ນ​ມາ ແລະ Node.js ກໍ​ໃຊ້ engine ນີ້​​ເອງ

* ແຕ່​ການ​ທີ່ Node.js ເປັນ​ທີ່​ນິຍົມ​ຂຶ້ນ​ມາ​ບໍ່​ແມ່ນ​ແຕ່ວ່າ​ມັນ​ມາ​ຖືກ​ທີ່​ຖືກ​ເວລາ​ເທົ່າ​ນັ້ນ​ ແຕ່​ເພາະ​ມັນ​ໄດ້​ສະແດງ​ໃຫ້​ເຫັນ​ແລ້ວ​ວ່າ ການ​ອອກ​ແບບ​ແລະ​ແນວ​ຄິດ​ຂອງ​ມັນ​ຊ່ວຍ​ນັກ​ພັດທະນາ​ທັງ​ຫຼາຍ​ໃຫ້​ສາມາດ​ໃຊ້ JavaScript ທາງ​ຝັ່ງ server ໄດ້​ງ່າຍ​ຂຶ້ນ​ຫລາຍ​ອີກດ້ວຍ​. (Nodejs ແມ່ນຫຍັງ [https://expert-programming-tutor.com/tutorial/article/A06\_NodeJS01\_HelloWorld.php 13 May 2017](https://expert-programming-tutor.com/tutorial/article/A06_NodeJS01_HelloWorld.php%2013%20May%202017)​).
  + - 1. ExpressJS

Express.js ເປັນ Web Application Framework ຊື່ດັ່ງທີ່ໄດ້ຮັບຄວາມນິຍົມຫຼາຍສຳ ລັບເຮັດວຽກເທິງ platform ຂອງ Node.js ຊຶ່ງເປັນ Server ຕົວໜຶ່ງໂດຍທັງ Express.js ແລະ Node.js ຕ່າງກໍໃຊ້ພາສາ JavaScript ໃນການພັດທະນາ ຖ້າເປັນ Web Application Framework ໃນສະໄໝກ່ອນ ຄົົນທີ່ພັດທະນາຈະຕ້ອງມີຄວາມຮູ້ຫຼາຍກວ່າ 1 ພາສາ, ພາສາທີ່ເຮັດວຽກທາງຝັ່ງ Server ຢ່າງ PHP ຫລື ASP ແລະ ພາສາທີ່ເຮັດວຽກທາງຝັ່ງ Client ຢ່າງ JavaScript เພື່ອລຸດຄວາມຫຍຸ້ງຍາກທັງໝົດເຖິງເວລາໃນການຕ້ອງຮຽນຮູ້ຫຼາຍໆ ພາສາເຮັດໃຫ້ເກີດ Node.js ກັບ Express.js ພຽງແຕ່ມີຄວາມຮູ້ JavaScript ກໍສາມາດຂຽນໄດ້ທັງ Server ແລະ Client ນອກຈາກນີ້ຖ້າໃຜເຄີຍຂຽນ JavaScript ຈະຮູ້ວ່າມັນມີການຕອບສະໜອງທີ່ວ່ອງໄວແນ່ນອນ ວ່າ Express.js ກໍຢົກເອົາມາເປັນຂໍ້ເດັ່ນໃນເລື່ອງຄວາມໄວ ໃນເລື່ອງການການຮຽນຮູ້ການຂຽນ Express.js ຈະໃຊ້ຮູບແບບທີ່ງ່າຍໃນການຮຽນຮູ້ຫລາຍທີ່ສຸດສຳລັບການ ພັດທະນາ Express.js ໃນເວັບໄຊຈະເວົ້າເຖິງການໃຊ້ Routing (ການກຳນົດເສັ້ນທາງຂອງລະບົບ) ແລະ Middleware (ການຮັບສົ່ງຂໍ້ມູນຂອງລະບົບ) ສາມາດຂຽນໄດ້ໃນຮູບແບບ MVC ສ່ວນການເຊື່ອມຕໍ່ກັບຖານຂໍ້ມູນສາມາດໃຊ້ MongoDB ຫລືຈະໃຊ້ MySql ກໍໄດ້ສຳລັບນາມສະກຸນຂອງໄຟລຄື .js ຂະນະນີ້ໄດ້ພັດທະນາມາເຖິງເວີຊັ່ນທີ່ 4.x ແລ້ວ.

* + - 1. Version Control (Git ແລະ GitHub)

ເນື່​ອງ​ຈາກ​ທຸກ​ມື້​ນີ້​ແມ່ນໃຊ້ Git ໃນ​ການ​ເຮັດ​ວຽກງານ​ຢູ່​ແລ້ວ ແລະ ​ບາງເທື່ອ​ກໍ​ຕ້ອງ​ສອນ​ເລື່ອງ​ນີ້​ໃຫ້​ກັບ​ສະມາຊິກ​ໃໝ່​ພາຍ​ໃນ​ທີມ ກໍ​ເລຍ​ຕັດ​ສິນ​ໃຈ​ຂຽນ​ບົດ​ຄວາມ​ນີ້​ຂຶ້ນ​ມາ​ ເພື່ອ​ທີ່​ຈະ​ໄດ້​ແຊ​ຄ​ວາມ​ຮູ້​ໃຫ້​ກັບ Developer ທີ່​ສົນ​ໃຈ​ໃນ Git ແຕ່​ຍັງ​ບໍ່​ເຂົ້າໃຈ​ ຫລື​ ບໍ່​ຮູ້​ວ່າ​ຈະ​ເຮັດ​ຄວາມ​ເຂົ້າໃຈ​ແບບໃດ​​ດີ ເພາະ​ທຸກ​ມື້​ນີ້​ໂລກ​ຂອງ Developer ນັ້ນ​ໄປ​ໄວ​ເຫຼືອ​ເກີນ ຫຼາຍ​ໆ​ຢ່າງ​ຖ້າ​ເຮົາ​ບໍ່​ຮູ້​ຈັກ ​ຫຼື ​ໃຊ້​ບໍ່​ເປັນ​ກໍ​ອາດຈະ​ເຮັດໃຫ້​ພາດ​ໂອ​ກາດ​ດີ​ໆ​ໃນ​ເສັ້ນ​ທາງ​ນີ້​ກໍ​ເປັນ​ໄດ້

1. **ຄວາມ​ສຳຄັນ​ຂອງ Version Control**

ລອງ​ຄິດ​ເບິ່ງ​ວ່າ​ຖ້າ​ຕ້ອງ​ພັດທະນາ​ໂປຣ​ເຈກ​ຂະໜາດ​ໃຫຍ່​ທີ່​ມີ Developer 4 ຄົົນທີ່​ກຳ​ລັງ​ມະ​ຣຸມ​ມະ​ຕຸ້ມ​ໂຄ້​ດ​ໃນ​ໂປຣ​ເຈກ​ນີ້​ຢູ່ ເຈົ້າ​ຈະ​ໃຊ້​ວິທີ​ໃດ​ເພື່ອ​ເອົາ​ໂຄ້​ດ​ທີ່​ແຕ່​ລະ​ຄົນ​ຂຽນ​ມາ​ທັງຫມົດ​ເຂົ້າ​ນຳກັນ​ໃນ​ໂປຣ​ເຈກ

ວິທີ​​ເກົ່າ​​ທີ່​ໃຊ້​ກັນ​ກໍ​ຄື ກ໋ອບ​ໂປຣ​ເຈກ​ຈາກ​ແຕ່​ລະ​ຄົນ​ມາ​ທັງຫມົດ​ໄວ້​ໃນ​ເຄື່ອງ​ດຽວ​ກັນ ແລ້ວ​ນັ່ງ​ລວມກັນ ມີ​ຄົນ​ໜຶ່ງ​ເຮັດ​ຫນ້າທີ່​ເປີດ​ໂຄ້​ດ​ຂອງ​ແຕ່​ລະ​ຄົນ​ຂຶ້ນ​ມາ ສົມໝຸດວ່າຄົນ​ໆ​ນັ້ນ​ຄື​ ທ້າວ A ແລະ​ ທ້າວ B ເປັນ​ຄົນ​ຂຽນ​ໂຄ້​ດ​ທີ່​ກຳ​ລັງ​ຈະ​ທັງຫມົດ​ໄວ້​ໃນ​ໂປຣ​ເຈກ​ດຽວ​ກັນ ທ້າວ A ກໍ​ເລີຍ​ຕ້ອງ​ຖາມ​ ທ້າວ B ວ່າ​ຂຽນ​ໂຄ້​ດ​ບ່ອນໃດ​ ແລ້ວ​ຄ່ອຍ​ເອົາ​ໄປ​ແປະ​ທັງຫມົດ​ໄວ້​ໃນ​ໂປຣ​ເຈັກ​ຫຼັກ

ບັນຫາ​ເກົ່າ​ໆ​ທີ່ເກິດຂື້ນ​ຢູ່​ປະຈຳ​ກໍ​ຄື ໂຄ້​ດ​ທີ່​ ທ້າວ B ໄປ​ແກ້​ໄຂ ພັດ​ໄປ​ທັບ​ຊ້ອນ​ກັບ​ ທ້າວ C ເພາະ​ທ້າວ C ກໍ​ແກ້​ໄຂ​ໂຄ້​ດ​ຈຸດ​ນັ້ນ​ຄືກັນ ກາຍ​ເປັນ​ວ່າ​ ທ້າວ B ກໍ​ຕ້ອງ​ໄປ​ເອີ້ນ​ ທ້າວ C ມາ​ນັ່ງຖາມ​ ເພື່ອ​ບອກ​ໃຫ້​ ທ້າວ A ແກ້​ໄຂ​ໃຫ້​ໂຄ້​ດ​ຂອງ​ ທ້າວ B ແລະ​ ທ້າວ C ເຮັດວຽກ​ຮ່ວມ​ກັນ​ໄດ້

ຍັງ​ບໍ່​ລວມ​ໄປ​ເຖິງ​ກໍລະນີ​ທີ່​ ທ້າວ B ຈື່​ບໍ່​ໄດ້​ວ່າ​ຕັວ​ເອງ​ແກ້​ໄຂ​ໂຄ້​ດ​​ຈຸດ​ໃດ​ແດ່ ເພາະ​ຟີ​ເຈີທີ່​ເຮັດ​ນັ້ນ​ໃຊ້​ເວລາ​ຫຼາຍ​ມື້​ ແລະ ​ຂຽນ​ໂຄ້​ດ​ຫຼາຍ​ບັນ​ທັດ ກາຍ​ເປັນ​ວ່າ​ຕ້ອງ​ນັ່ງ​ລວມກັນ​ເພື່ອ​ໄລ່​ເບິ່ງ​ໂຄ້​ດວ່າ​ພາດ​ຈຸດ​ໃດ​ຫລື​ ບໍ່ ຊຶ່ງ​ລວມປ​ເຖິງ​ບັນຫາ​ອື່ນ​ໆ​ທີ່​ຈະ​ຕາມ​ມາ​ອີກ​ຫລວງ​ຫລາຍ

ສະນັ້ນ​ ຈຶ່ງເຮັດໃຫ້ເກີດຂຶ້ນ​ທີ່​ເອີ້ນວ່າ Version Control ຂຶ້ນ​ມາ​ເພື່ອ​ຄວບ​ຄຸມ​ການ​ປ່ຽນ​ແປງ​ຂອງ​ໂຄ້​ດ​ໃນ​ໂປຣ​ເຈກ ໂດຍ​ປະໂຫຍດ​ທີ່​ເຫັນ​ໄດ້​ແຈ້ງ​ຂອງ Version Control ຈະ​ມີ​ດັ່ງ​ນີ້:

* ເກັບ​ປະຫວັດ​ການ​ແກ້​ໄຂ​ໂຄ້​ດ​ໄວ້​ທຸກ​ເທື່ອ ແລະ​ ຮູ້​ໄດ້​ວ່າ​ໂຄ້​ດ​ຈຸດໃດ​ໃຜ​ເປັນ​ຄົນ​ເພີ່ມ​ເຂົ້າ​ມາ​ ຫຼື ​ແກ້​ໄຂ
* ຊ່ວຍ​ລວມໂຄ້​ດ​ຈາກ​ຫຼາຍ​ໆ​ຄົນ​ເຂົ້າ​ນຳກັນ​ໃຫ້​ງ່າຍ​ຂຶ້ນ ເບິ່ງ​ໄດ້​ວ່າ​ໂຄ້​ດ​ເດີມ​ຄື​ຫຍັງ ແລະ ​ແກ້​ໄຂ​ເປັນ​ຫຍັງ
* ເມື່ອ​ເກີດ​ບັນຫາ​ກໍ​ສາມາດ​ຕິດ​ຕາມ​ເບິ່ງ​ປະຫວັດ​ການ​ແກ້​ໄຂ​ໂຄ້​ດ​ໃນ​ແຕ່​ລະ​ໄຟ​ລ​ ​ແຕ່​ລະ​ບັນ​ທັດ​ໄດ້​ງ່າຍ
* ຊ່ວຍ​ໃຫ້​ສາມາດ​ຈັດການ​ໂປຣ​ເຈກ​ໄດ້​ຢ່າງ​ເປັນ​ ເບິ່ງ​ໂຄ້​ດ​ແຕ່​ລະ​ສ່ວນ​ເປັນ​ຟີ​ເຈີ ບໍ່​ຂຽນ​ໂຄ້​ດ​ຂ້າມ​ຟີ​ເຈີ​ໄປ​ມາ​ໃນ​ໂຄ້​ດ​ຊຸດ​ດຽວ​ກັນ
* ເປັນ Backup ໄປ​ໃນ​ຕັວ ບໍ່​ຕ້ອງ​ຢ້ານ​ເວລາ​ໂຄ້​ດ​ມີ​ບັນຫາ​ແລ້ວ​ຕ້ອງ Rollback ກັບ​ໄປ​ໃຊ້​ໂຄ້​ດ​ຊຸດ​ເກົ່າ ແລະ ​ໃຊ້​ພື້ນ​ທີ່​ໃນ​ການ​ເກັບ​ຂໍ້​ມູນ​ນ້ອຍ​ເມື່ອ​ທຽບ​ກັບ​ການ Backup ແບບ​ເກັບ​ທັງ​ໂປຣ​ເຈກ​ໄວ້​ທຸກ​ເທື່ອ​ບ່ອນ​ເຮັດວຽກ Backup
* ສາມາດ Track ການ​ເຮັດ​ວຽກງານ​ຂອງ​ທຸກ​ຄົນ​ພາຍ​ໃນ​ທີມ​ໄດ້​ຈາກ History

1. ການເຮັດວຽກຂອງ Git

ທີ່ຈິງແລ້ວມັນ​ກໍ​ຄື​ການ​ເຮັດ​ວຽກງານ​ແບບ Distributed Version Control .ໃຫ້​ລອງ​ຄິດເຖິງ​ພາ​ບວ່າ​ມີ Server ກາງ​ເຄື່ຶອງ​ໜຶ່ງ​ທີ່​ລໍ​ຖ້າ​ເກັບ​ຂໍ້​ມູນ​ຈາກ​ຜູ້​ໃຊ້​ແຕ່​ລະ​ເຄື່ອງ​ກ່ອນ​

Server ກາງ​ເອີ້ນວ່າ Remote 1 ສ່ວນ Developer ທີ່​ເຮັດ​ໂປຣ​ເຈກ​ນີ້​ຢູ່​ຈະ​ຮຽກວ່າ Computer 1 ແລະ 2 ຊຶ່ງ​ແຕ່​ລະ​ຄົນ​ກໍ​ເຮັດ​ຄົນ​ລະ Feature ຢູ່ ຊຶ່ງ Feature ຂອງ​ແຕ່​ລະ​ຄົນ​ນັ້ນ​ກໍ​ຈະ​ຖືກ​ເກັບ​ໄວ້​ທີ່ Remote 1, ອາດຈະ​ເບິ່ງ​ຄື​ວ່າ​ມັນ​ເປັນ​ລະບົບ Server ກາງ​ທີ່​ຄອຍ​ຖ້າ Backup ຂໍ້​ມູນ​ຂອງ​ທຸກ​ຄົນ​ຢູ່​ຕະຫລອດ​ເວລາແຕ່​ໃນ​ຄວາມ​ເປັນ​ຈິງ Git ນັ້ນ​ມີ​ຫຍັງ​ຫລາຍກວ່າ​ນີ້​ອີກ​ຫຼາຍຢ່າງ, Git ອອກ​ແບບ​ມາ​ໃຫ້​ເຮັດວຽກ​ກະ​ຈາຍ​ແບບ​ບໍ່​ມີ​ສູນ​ກາງ ທຸກ​ເຄື່ອງ​ເຮັດວຽກ​ເປັນ VCS ດ້ວຍ​ຕັວ​ເອງ​ໄດ້. ນັ້ນ​ໝາຍ​ຄວາມ​ວ່າ​ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ມີ Server ກາງ​ກໍ​ໄດ້ ສາມາດ​ໃຊ້​ເຄື່ອງ​ສ່ວນ​ຕົວ​ເຮັດ​ເປັນ VCS ໄດ້​ເລີຍ ແຕ່​ຖ້າ​ຕ້ອງ​ເຮັດວຽກ​ຮ່ວມ​ກັນ​ຫຼາຍ​ໆ​ເຄື່ອງ ກໍ​ຕ້ອງ​ໃຊ້ Server ເປັນ​ຕົວ​ກາງ​ໃນ​ການ​​ຂໍ້​ມູນ, ຊຶ່ງ​ຈະ​ເຮັດໃຫ້​ຂໍ້​ມູນ​ຂອງ​ເຮົາ​ບໍ່​ຜູກ​ຂາດ​ກັບ Server ຈົນ​ເກີນ​ໄປ ໃນ​ເວລາ​ທີ່ Server ກາງ​ມີ​ບັນຫາ​ຫລື​ ວ່າ​ເຮັດວຽກ​ແບບ Offline ເຮົາ​ກໍ​ຍັງ​ຄົງ​ເຮັດວຽກ​ໄດ້​ຢູ່ ໂດຍ​ໃຊ້​ຂໍ້​ມູນ​ຈາກ VCS ພາຍ​ໃນ​ເຄື່ອງ​ຕົວ​ເອງ ພໍ​ເຊື່ອມ​ຕໍ່​ກັບ Server ກາງ​ກໍ​ຄ່ອຍ Sync ຂໍ້​ມູນ​ພາຍ​ຫລັງ​ໄດ້ ແລະ​ ເມື່ອ​ທຸກ​ໆ​ເຄື່ອງ​ເຮັດວຽກ​ເປັນ VCS ຢູ່​ແລ້ວ ຈຶ່ງ​ເຮັດໃຫ້​ເຮົາ​ສາມາດ​ມີ Server ກາງ​ຫລາຍກວ່າ 1 ເຄຶ່ອງ​ໄດ້​ເຊັ່ນ​ກັນ

1. Server ສຳລັບໃຫ້ບໍລິການ Git (Version Control Repository Hosting Service)

ການໃຊ້ Git ໃຫ້ເກີດປະໂຫຍດທີ່ສຸດຕ້ອງມີ Server ທີ່ເຮັດຫນ້າທີ່ເປັນ VCS ຊຶ່ງໃນປະຈຸບັນນີ້ກໍມີທາງເລືອກຫລວງຫລາຍບໍ່ວ່າຈະໃຊ້ບໍລິການຈາກເວັບຕ່າງໆ ຫລື ຕັ້ງ Server ຂຶ້ນມາເອງ

ໃນກໍລະນີເລີ່ມຕົ້ນ ຂໍແນະນຳໃຫ້ໃຊ້ Server ທີ່ໃຫ້ບໍລິການ Git ໄປເລີຍ ເພາະເຮົາບໍ່ຕ້ອງຈັດການຫຍັງຫລາຍ ພຽງແຕ່ສະໝັກກໍເຂົ້າໄປໃຊ້ງານໄດ້ເລີຍ ໂດຍເວັບໄຊທີ່ນິຍົມກໍຈະມີດັ່ງນີ້: GitHub, Bitbucket, Gitlab. ຊຶ່ງເວັບເຫຼົ່ານີ້ຈະມີໃຫ້ບໍລິການທັງແບບ Public (ໃຜໆກໍເຂົ້າມາເບິ່ງໂຄ້ດຂອງເຮົາໄດ້)ແລະ Private (ສະເພາະຄົົນທີ່ເຮົາອະນຸຍາດເທົ່ານັ້ນ) ຖ້າຢາກເຮັດເປັນ Open Source ຢູ່ແລ້ວ ກໍແນະນຳ GitHub

# ຄຳ​ຕ່າງ​ໆ​ທີ່​ຈະ​ຕ້ອງ​ຮູ້​ຈັກ​ເມື່ອ​ໃຊ້​ງານ Git

**ກ. Repository**

ເວລາ​ທີ່​ເຮົາ​ພັດທະນາ​ໂປຣ​ແກຣມ​ ເຮົາ​ຈະ​ຕ້ອງ​ສ້າງ​ສິ່ງ​ທີ່ເອີ້ນວ່າ Project ຊຶ່ງ​ການສ້າງ Project ສຳລັບ​ໃຊ້​ງານ Git ຈະ​ເອີ້ນວ່າ Repository

ຊຶ່ງ Repository ຂອງ Git ແທ້ໆ​ແລ້ວ​ກໍ​ຄື Folder ທີ່​ໃຊ້​ເກັບ​ຂໍ້​ມູນ​ນັ້ນ​ເອງ ຢາກ​ຈະ​ເກັບ​ຫຍັງ​ໄວ້​ໃນ​ນັ້ນ​ກໍ​ຍັດ​ເຂົ້າໄປ​ໄດ້​ເລີຍ (ໃນ​ຄວາມ​ເປັນ​ຈິງ 1 Repository ສາມາດ​ເກັບ Project ເທົ່າໃດ​ກໍ​ໄດ້​ຕາມ​ທີ່​ຕ້ອງ​ການແຕ່​ສ່ວນ​ໃຫ່ຍ​ກໍ​ນິຍົມ​ເກັບ Project 1 ຕົວ​ຕໍ່ 1 Repository)

ເວລາ​ທີ່​ຜູ້​ອ່ານ​ເປີດ​ເບິ່ງ​ໂຄ້​ດ​ຕ່າງ​ໆ​ຂອງ​ນັກ​ພັດທະນາ​ເທິງ GitHub (ຍົກ​ຕົວ​ຢ່າງ​ເຊັ່ນ<https://github.com/google/volley>) ໃນ​ແຕ່​ລະ​ໂປຣ​ເຈກ​ທີ່​ເຮົາ​ເຫັນ​ ທີ່​ເອີ້ນວ່າ Repository

## **ຂ. Clone**

ເວລາ​ທີ່​ຜູ້​ອ່ານ​ມີ Repository ຢູ່​ເທິງ Remote ​ຢູ່​ແລ້ວ ແລະ​ ຕ້ອງ​ການ Sync ມາ​ລົງ​ເຄື່ອງ​ຂອງ​ເຮົາ ເຮົາ​ຈະ​ຕ້ອງ​ເຮັດ​ສິ່ງ​ທີ່​ເອີ້ນວ່າ CloneRepository ຫລື​ ກໍ​ຄື​ການ​ກ໋ອບ Repository ຈາກ Remote ມາ​ລົງ​ເຄື່ອງ​ນັ້ນ​ເອງ

## **ຄ. Commit**

ເວລາ​ທີ່​ມີ​ຂໍ້​ມູນ​ທີ່​ແກ້​ໄຂ​ສຳເລັດ​ແລ້ວ (ໂຄ້​ດ​ທີ່​ຂຽນ​ຄຳ​ສັ່ງ​ບາງຢ່າງ​ສຳເລັດ​ແລ້ວ) ແລ້ວ​ຢາກ​ຈະ​ກໍ່ການ Backup ເກັບ​ໄວ້​ໃນ VCS ຈະ​ຮຽກ​ກັນ​ວ່າ Commitຊຶ່ງ​ການ Commit ຈະ​ສາມາດ​ເລືອກ​ໄດ້​ວ່າ​ຈະ​ເອົາ​ໄຟ​ລ​​ໃດ​ອີກ (ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເລືອກ​ທຸກ​ໄຟ​ລ​)

ຊຶ່ງ​ເບື້ອງ​ຫຼັງ​ຂອງ Commit ກໍ​ຄື​ໃນ​ແຕ່​ລະ​ເທື່ອ​ທີ່​ເຮັດວຽກ Commit ມັນ​ຈະຈື່ແຕ່ວ່າ​ມີ​ຈຸດໃດ​ຂອງ​ຂໍ້​ມູນ​ທີ່​ຖືກ​ປ່ຽນ​ແປງ​ໄປ​ເລື້ອຍ​ໆ ດັ່ງ​ນັ້ນ​ໃນ​ແຕ່​ລະ Commit ມັນ​ຈະ​ບໍ່​ມີ​ໄຟ​ລ​​ຂໍ້​ມູນ​ສະບັບ​ເຕັມ ແຕ່​ມັນ​ສາມາດ​ຍ້ອນ​ເບິ່ງ History ໄດ້​ວ່າ​ມີ​ການ​ແກ້​ໄຂ​ຫຍັງ​ແດ່ ເຮັດໃຫ້​ຮູ້​ວ່າ​ໃນ Commit ນັ້ນ​ໆ ແຕ່​ລະ​ໄຟ​ລ​​ມີ​ຂໍ້​ມູນ​ເປັນ​ແບບໃດ

ແລະ ​ໃນ​ການ Commit ແຕ່​ລະ​ເທື່ອ ຈະ​ຕ້ອງ​ໃສ່ Commit Message ເພື່ອ​ອະທິບາຍ​ລາຍ​ລະອຽດ​ຂອງ​ຂໍ້​ມູນ​ໃນ Commit ນັ້ນ​ໆ​ວ່າ​ເຮົາ​ເຮັດ​ຫຍັງ​ໄປ​ແດ່ ເພື່ອ​​ມາ​ເບິ່ງ​ໃນ​ພາຍ​ຫຼັງ​ຈະ​ໄດ້​ອ່ານ​ຈາກ Commit Message ໄດ້​ເລິຍ ບໍ່​ຕ້ອງ​ໄປ​ນັ່ງ​ກົດ​ເບິ່ງ​ເອງ​ເທື່ອ​ລະ​ໄຟ​ລ​ ດັ່ງ​ນັ້ນ​ການ Commit ທີ່​ດີ​ຈຶ່ງ​ຄວນ​ໃສ່​ໃຈ​ກັບ Commit Message ນຳອີກ

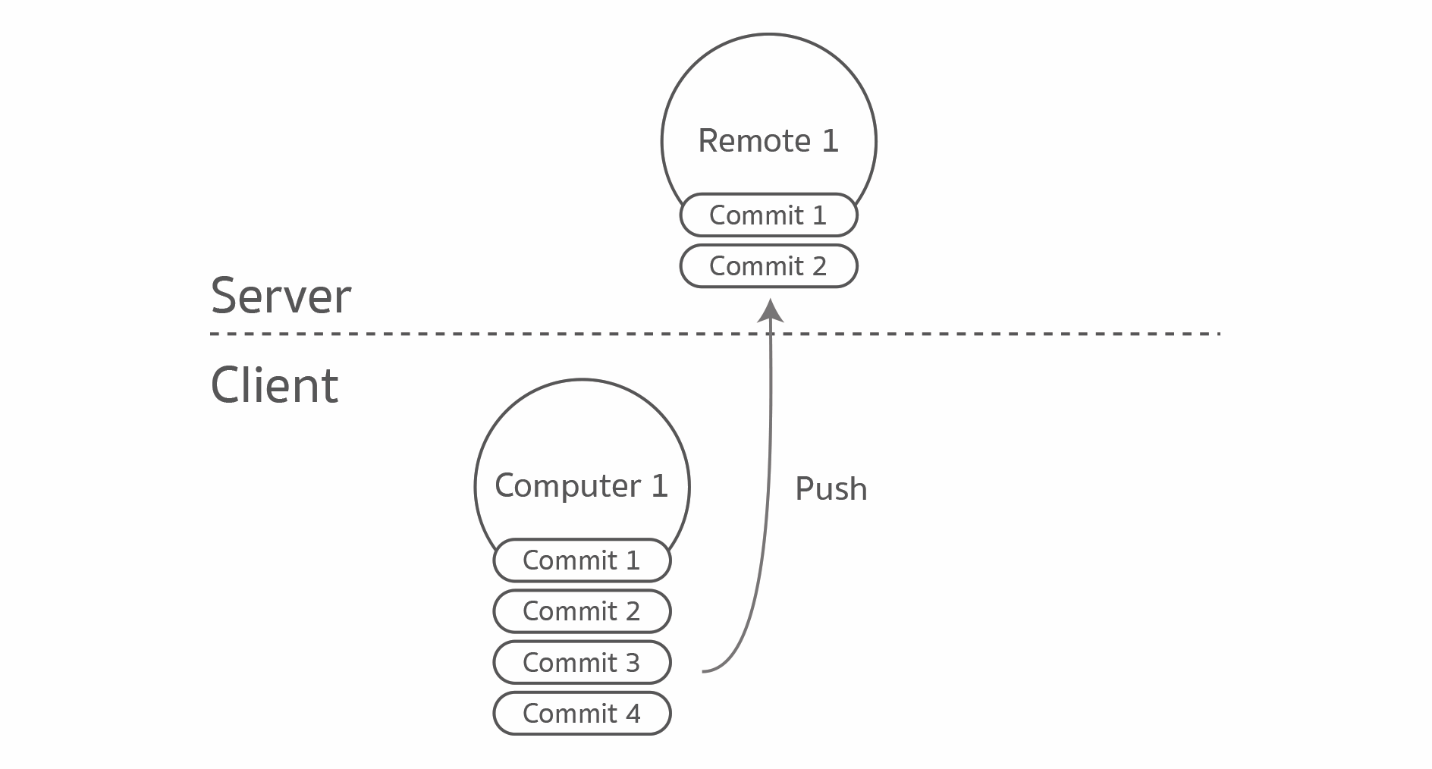
## **ງ. Unstaged ແລະ Staged**

ຂໍ​ເວົ້າ​ເຖິງ 2 ຄຳ​ນີ້​ໄປ​ພ້ອມ​ໆ​ກັນ​ເລີຍ

ເວລາ​ເຮົາ​ແກ້​ໄຂ​ໂຄ້​ດ​ ຫລື ​ແກ້​ໄຂ​ຂໍ້​ມູນ ໄຟ​ລ​ທີ່​ຖືກ​ແກ້​ໄຂ​ຈະ​ຢູ່ໃນ​ສ​ຖາ​ນະ Unstagedແລະ​ ເວລາ​ທີ່​ເຮົາ​ເຮັດ​ຫຍັງ​ສຳເລັດ​ຮຽບຮ້ອຍ ແລ້ວ​ຢາກ​ຈະ Commit ເກັບ​ໄວ້ ຈະ​ຕ້ອງ​ເລືອກ​ໄຟ​ລ​​ທີ່​ຕ້ອງ​ການ​ເພື່ອ​ຍ້າຍ​ເຂົ້າ​ສູ່​ໃນ​ສ​ຖາ​ນະ Stagedກ່ອນ​ເຖິງ​ຈະ​ກໍ່ການ Commit ໄດ້

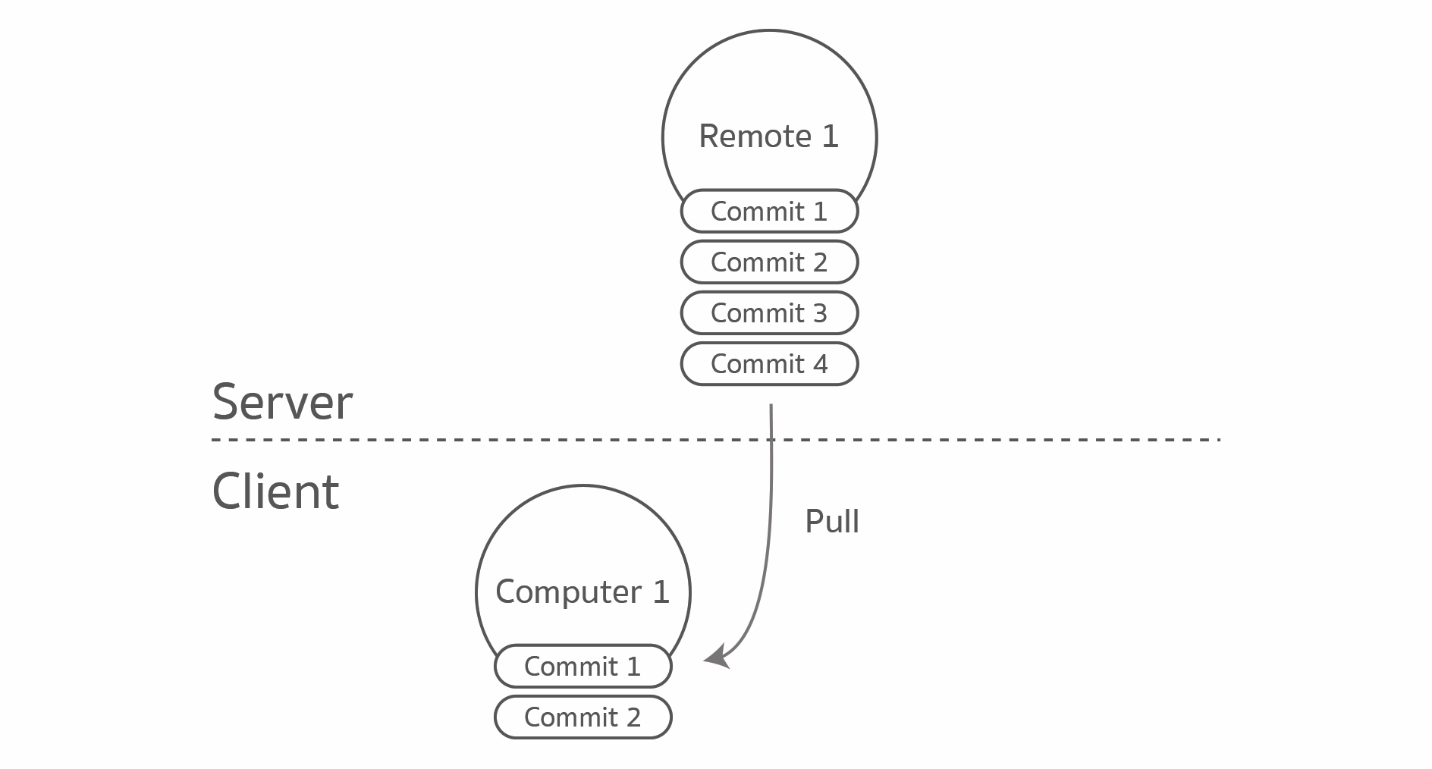
ຊຶ່ງ​ສະ​ຖາ​ນະ Unstaged ແລະ Staged ເຮັດໃຫ້​ເຮົາ​ສາມາດ​ເລືອກ​ສະເພາະ​ບາງ​ໄຟ​ລ​​ສຳລັບ Commit ໄດ້​ນັ້ນ​ເອງ ຈະ​ໄດ້ Commit ສະເພາະ​ໄຟ​ລ​​ທີ່​ເຮົາ​ຂຽນ​ສຳເລັດ​ແລ້ວ

**ຈ. Push**

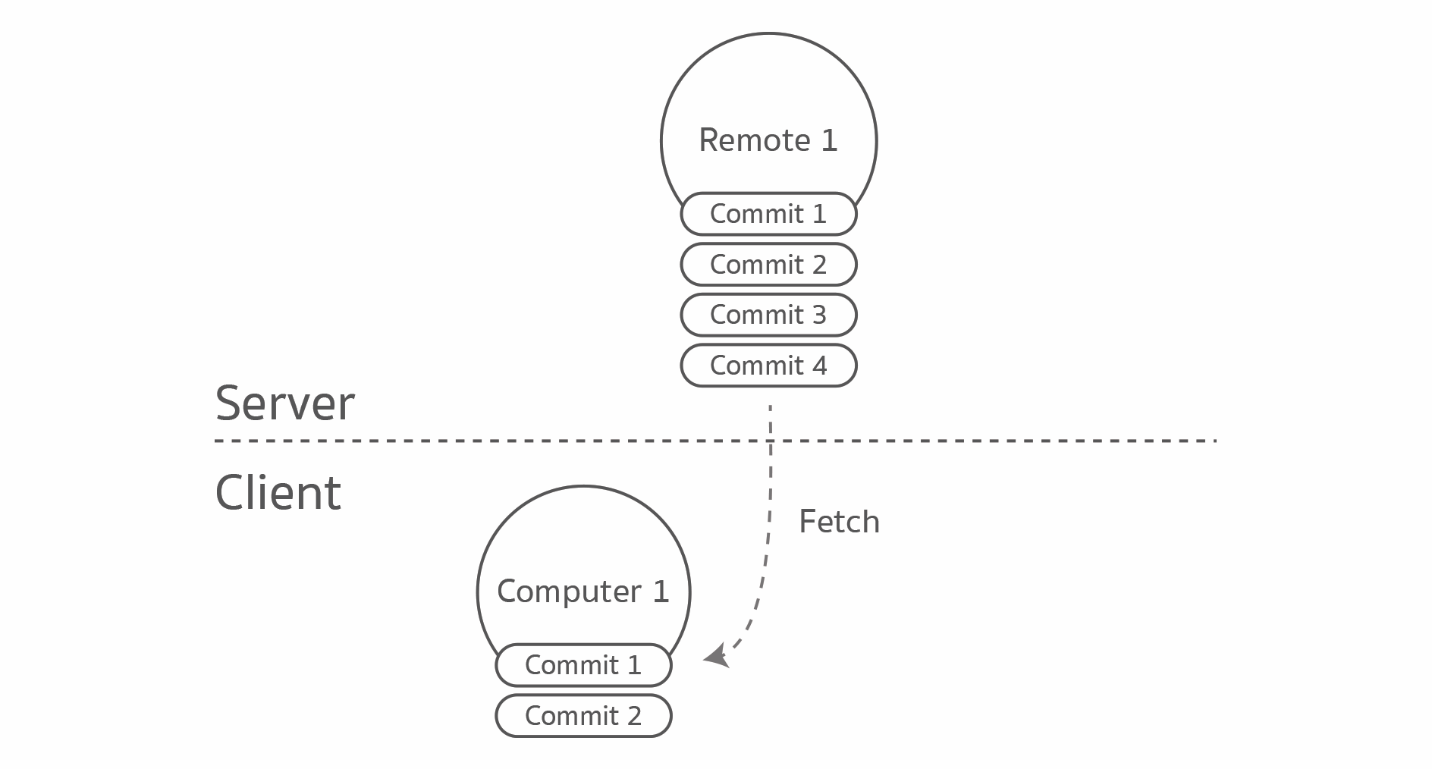
ເວລາ​ທີ່​ມີ Commit ຢູ່ໃນ​ເຄື່ອງ​ ແລ ະ​ຕ້ອງ​ການ​ຈະ Sync ຂຶ້ນ​ໄປ​ເກັບ​ໄວ້​ໃນ Remote ຈະ​เอี้นຂັ້ນ​ຕອນ​ນີ້​ວ່າ Push

**ສ. Pull**

ເວລາ Sync ຈາກ Remote ເພື່ອ​ດຶງ​ຂໍ້​ມູນ Commit ໃໝ່​ໆ​ລົງ​ມາ​ເກັບ​ໄວ້​ໃນ​ເຄື່ອງ​ຈະ​ຮຽກ​ຂັ້ນ​ຕອນ​ນີ້​ວ່າ Pull

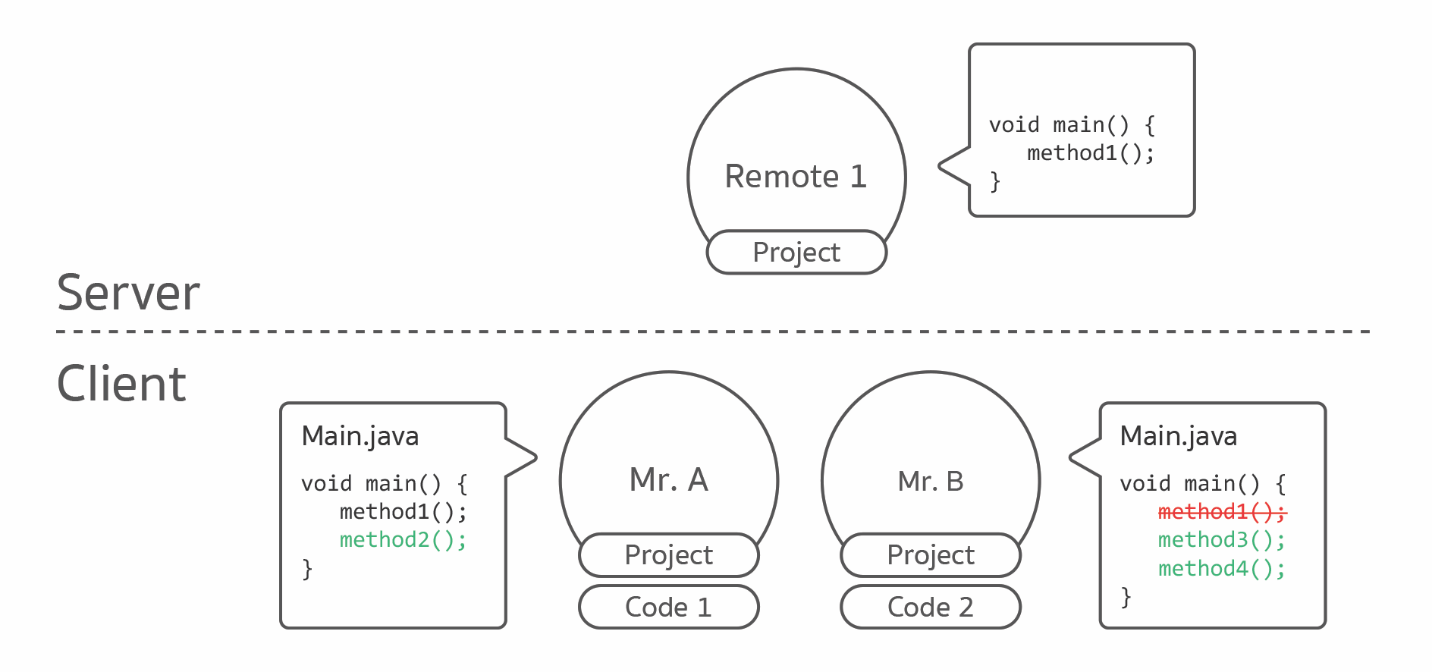


## **ຊ. Fetch**

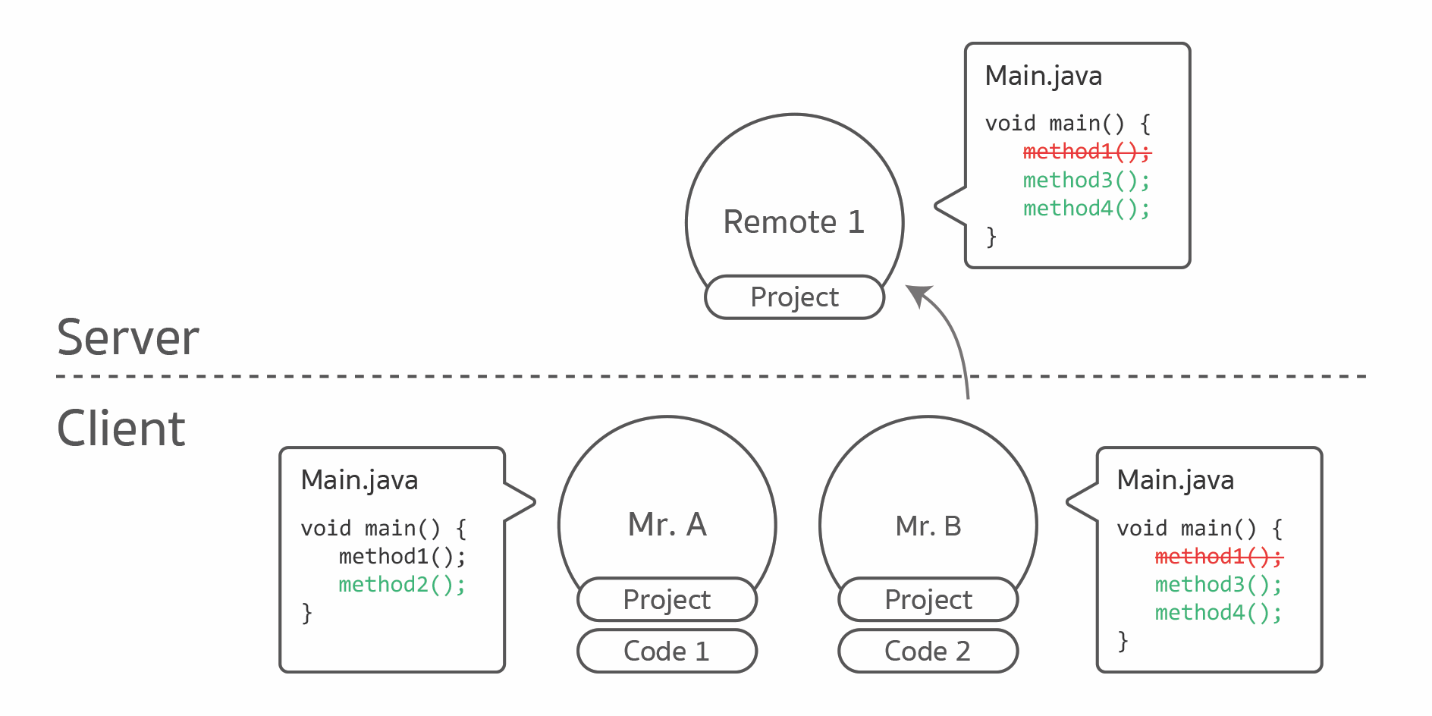
ໃນ​ບາງເທື່ອ​ເຮົາ​ອາດຈະ​ບໍ່​ຕ້ອງ​ການ Pull ຂໍ້​ມູນ​ລົງ​ມາ​ເກັບ​ໄວ້​ໃນ​ເຄື່ອງ​ທັນ​ທີ ແຕ່​ຢາກ​ເຊັກ​ສະ​ຖາ​ນະ​ຂອງ Remote ຊື່ໆ​ວ່າ​ມີ​ໃຜ Push ຂໍ້​ມູນ​ໃໝ່​ຂຶ້ນ​ໄປ​ທີ່ Remote ຫຼື​ບໍ່ ເຮົາ​ເອີ້ນວິທີ​ນີ້ວ່າ Fetch ຊຶ່ງ​ການ Fetch ຈະ​ເຮັດໃຫ້​ເຮົາ​ສາມາດ​ອັບ​ເດດ​ ແລະ​ ເບິ່ງ History ທັງ​ໝົດ​ທີ່​ຢູ່​ເທິງ Remote ໄດ້ໂດຍ​ບໍ່​ຕ້ອງ​ດຶງ​ຂໍ້​ມູນ​ລົງ​ມາ. ຊຶ່ງ​ທີ່ຈິງ​ແລ້ວ​ຂັ້ນ​ຕອນ​ການ Fetch ຈະ​ຖືກ​ເອີ້ນ​ທຸກ​ທີ່​ເຮັດວຽກ Pull. ດັ່ງ​ນັ້ນ ເຮົາ​ຈຶ່ງ​ສາມາດ​ເລືອກ​ໄດ້​​ວ່າ​ຈະ Pull ເລິຍ (​ມັນ​ກໍຈະ Fetch ເພື່ອ​ເຊັກ​ເອງ​) ຫຼື​ ຈະ Fetch ເບິ່ງ​ກ່ອນ​ວ່າ​ມີ Commit ຫຍັງ​ເພີ່ມ​ເຂົ້າ​ມາແລ້ວ​ຄ່ອຍ Pull ລົງ​ມາ

## **ຍ. Merge Commit**

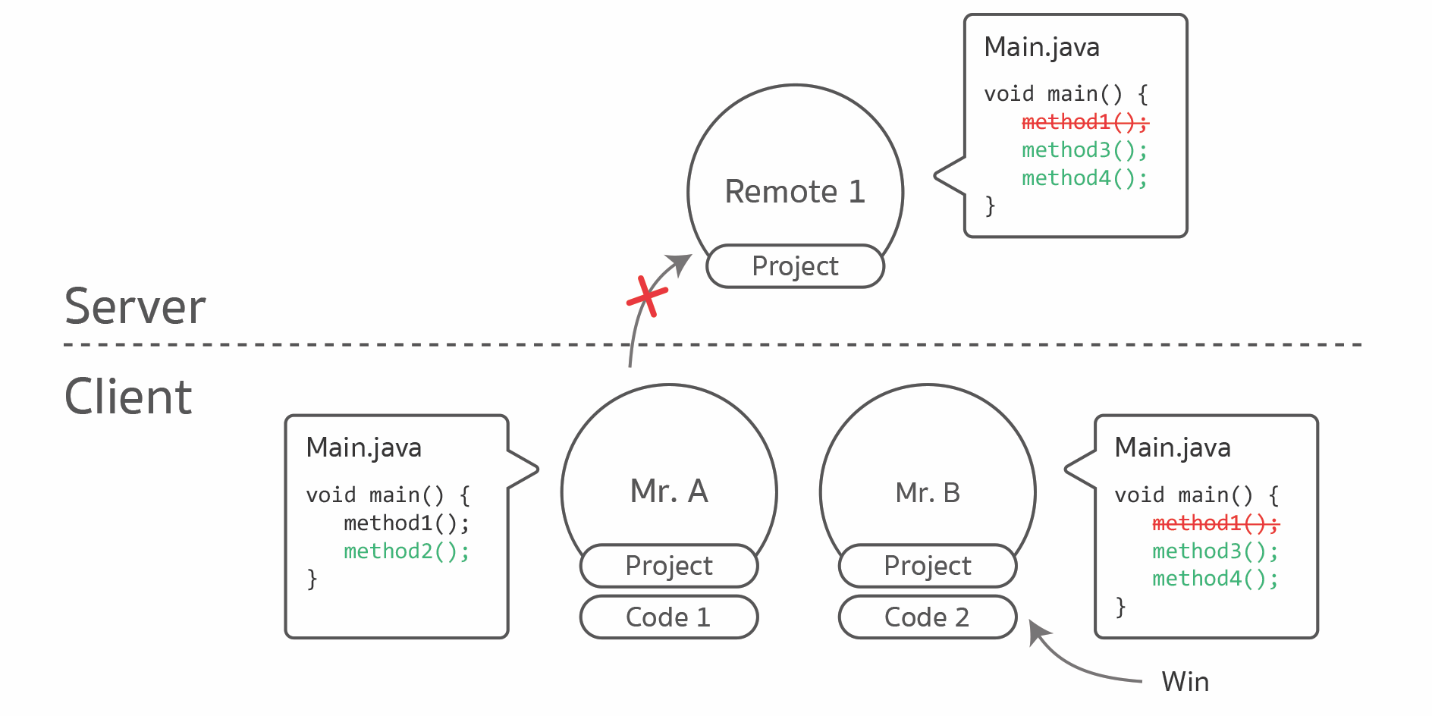
ສົມໝຸດ​​ວ່າ​ ທ້າວ A ກັບ​ ທ້າວ B ຂຽນ​ໂຄ້​ດ​ນຳກັນ​ຢູ່ ແລະ ​ທັງ​ຄູ່​ກໍ​ຂຽນ​ໂຄ້​ດ​ທີ່​ຢູ່ໃນ​ໄຟ​ລ​​ດຽວ​ກັນ



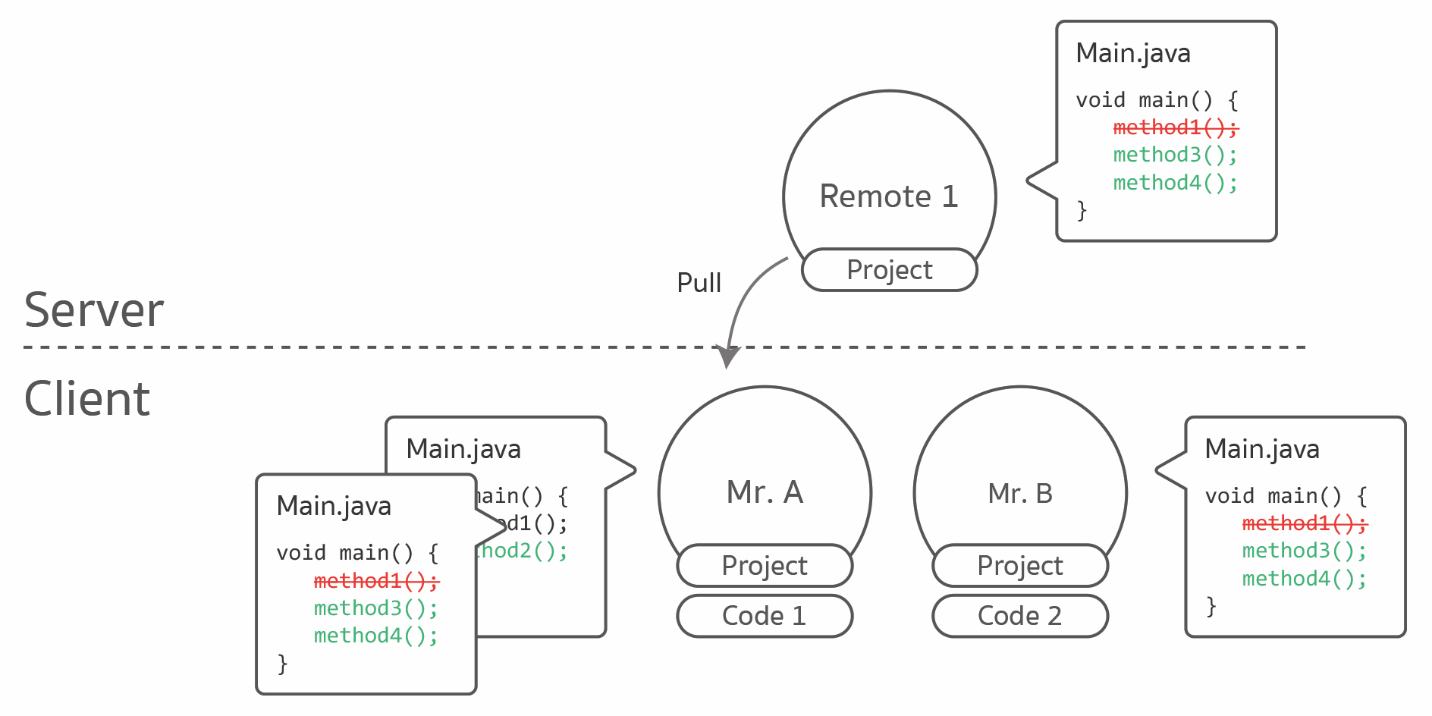
ທ້າວ B ຂຽນ​ໂຄ້​ດ​ສຳເລັດ​ແລ້ວ Commit ໄຟ​ລ​ທີ່​ຕົວ​ເອງ​ຂຽນ​ສຳເລັດ​ແລ້ວ Push ຂຶ້ນ Remote



ທ້າວ A ຂຽນ​ໂຄ້​ດ​ສຳເລັດ​ພາຍ​ຫລັງ ແລະ​ຈະ Push ຂຶ້ນ Remote ແຕ່ພົບວ່າ​ທ້າວ B ໄດ້ Push ຂຶ້ນ​ໄປ​ກ່ອນ​ແລ້ວ ຈຶ່ງ Push ຂຶ້ນ​ໄປ​ທັນ​ທີ​ບໍ່​ໄດ້



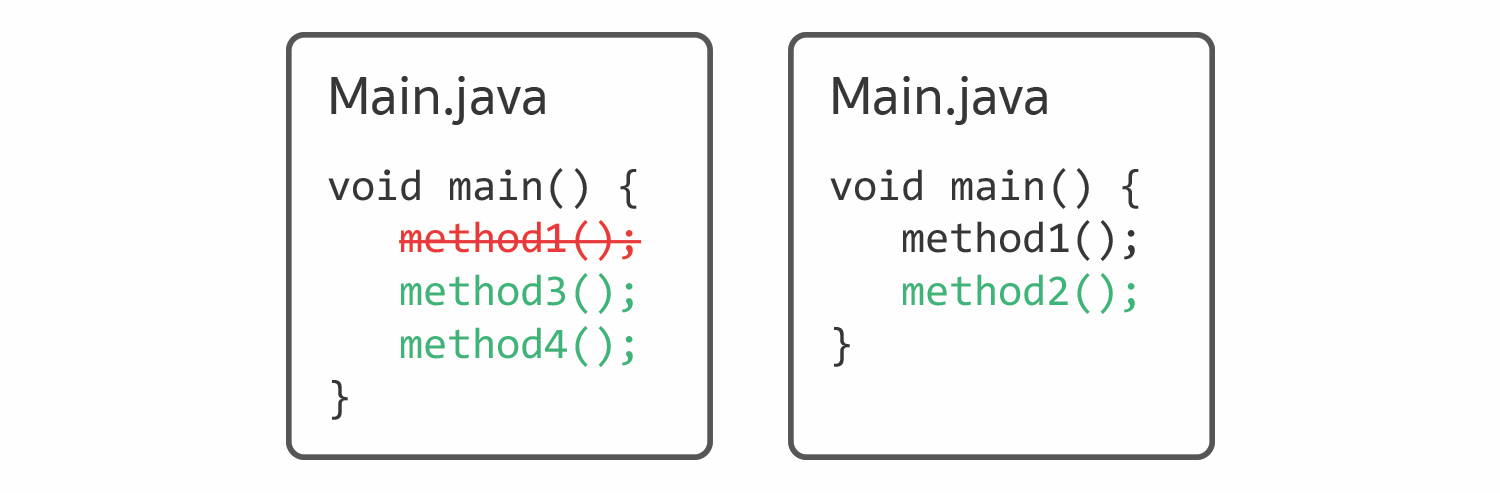
ດັ່ງ​ນັ້ນ​ສິ່ງ​ທີ່​ທ້າວ A ຕ້ອງ​ເຮັດ​ກ່ອນທີ່ຈະ Push ຂອງ​ຕົວ​ເອງ​ຂຶ້ນ​ໄປ​ໄດ້ກໍ​ຄື​ຈະ​ຕ້ອງ Pull ຈາກ Remote ລົງ​ມາ​ໃໝ່​ກ່ອນ​ເພື່ອ​ອັບ​ເດດ Commit ທີ່​ທ້າວ B ໄດ້ Push ຂຶ້ນ​ໄປ ຊຶ່ງ​ເຮົາ​ເອີ້ນຂັ້ນ​ຕອນ​ນີ້​ວ່າ Merge Commitນັ້ນ​ເອງ



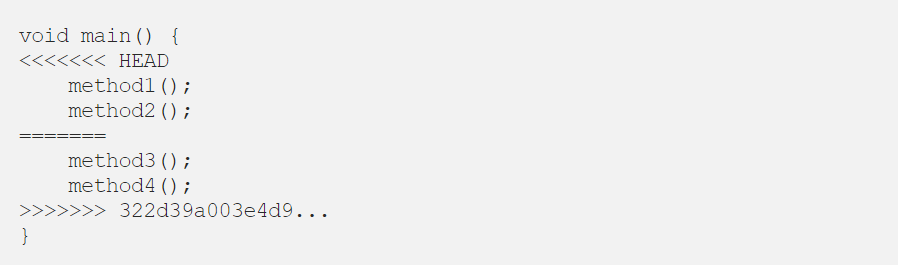
**ເພິ່ມເຕີມ** : ທີ່ຈິງແລ້ວ Git ສາມາດ Auto Merge ໄດ້​ເລຍ (ກົດ Pull > Merge ໄຟ​ລ​​ > Commit ໃຫ້​ທັນ​ທີ) ແຕ່​ກໍ​ແນະ​ນຳ​ວ່າ​ໃຫ້ Manual Merge ດີກວ່າ ເພື່ອ​ທີ່​ວ່າ​ຈະ​ໄດ້ Review Code ໄປ​ໃນ​ຕົວ​

## **ດ. Conflict**

ໃນ​ຂະນະ​ທີ່​ກຳ​ລັງ​ຈະ Merge ຢູ່​ນັ້ນຈະ​ເຫັນ​ວ່າ​ໂຄ້​ດ​ຂອງ​ທ້າວ A ແລະ​ ທ້າວ B ມັນ​ຕໍາ​ກັນ ໄປ​ແກ້​ໂຄ້​ດ​ຍ່ອນດຽວ​ກັນ ດັ່ງ​ນັ້ນ Git ຈະ​ແຈ້ງ​ວ່າ​ເກີດ Conflictຫລື ​ກໍ​ຄື​ໂຄ້​ດ​ທີ່​ທັບ​ຊ້ອນ​ກັນ​ນັ້ນ​ເອງ

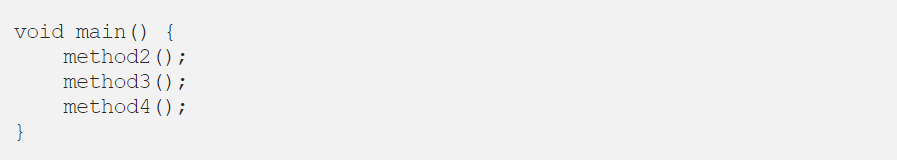


ຊຶ່ງ​ທ້າວ A ກໍ​ຕ້ອງ​ແກ້ Conflict ນີ້​ໃຫ້​ຮຽບຮ້ອຍ​ ເຖິງ​ຈະ Merge Commit ແລ້ວ Push ຂຶ້ນ Remote ໄດ້ ຊຶ່ງ​ຕອນ​ທີ່​ເກີດ Conflict ຂຶ້ນ ແລ້ວ​ເປີດ​ໂຄ້​ດ​ດັ່ງ​ກ່າວ​ມາ​ເບິ່ງ ກໍ​ຈະ​ເຫັນ​ເປັນ​ແບບ​ນີ້



ໃຫ້​ສັງເກດ​ຮູບ​ແບບ​ຂອງ​ເຄື່ອງ​ໝາຍ <<<, === ແລະ >>> ໃຫ້​ດີ​ໆ ແລ້ວ​ຈະ​ເຫັນ​ວ່າທີ່ຈິງແລ້ວ ​ມັນ​ມີ​ຮູບ​ແບບ​ທີ່​ເຂົ້າໃຈ​ໄດ້​ງ່າຍ​ຫລາຍໂດຍ​ທີ່

* ໂຄ້​ດ​ທີ່​ຢູ່​ລະຫວ່າງ <<< ແລະ === ຄື​ໂຄ້​ດ​ຂອງ​ທ້າວ A
* ໂຄ້​ດ​ທີ່​ຢູ່​ລະຫວ່າງ === ແລະ >>> ຄື​ໂຄ້​ດ​ຂອງ​ທ້າວ B ສ່ວນ​ຕົວ​ເລກ​ຕໍ່​ທ້າຍ​ຄື​ໝາຍ​ເລກ​ຂອງ Commit ບ່ອນ​ເຮັດວຽກ Merge

ເພື່ອ​ໃຫ້​ໂຄ້​ດ​ເຮັດວຽກ​ໄດ້​ເໝາະ​ສົມ ທ້າວ A ກໍ​ຈະ​ຕ້ອງ​ເອົາ​ໂຄ້​ດ​ທີ່​ຕົວ​ເອງ​ແກ້​ໄຂ ໄປ​ທັງຫມົດ​ກັບ​ໂຄ້​ດ​ຂອງ​ທ້າວ B ໃຫ້​ເຮັດວຽກ​ໄດ້

ເມື່ອ​ສຳເລັດ​ແລ້ວ​ກໍ​ໃຫ້ Commit (ຊຶ່ງ​ຈະ​ເປັນ​ການ Merge Commit) ແລ້ວ​ທ້າວ A ກໍ​ຈະ​ສາມາດ Push ຂຶ້ນ Remote ໄດ້​ແລ້ວ (ທ້າວ B ກໍ​ຈະ​ຕ້ອງ Pull Commit ພາຍ​ຫລັງ ແລະ ​ຖ້າ​ຫຼົງ​ໄປ​ແກ້​ໄຂ​​ຊ້ຳ​ກໍ​ອາດຈະ​ເກີດ Conflict ໄດ້)

Conflict ຖື​ວ່າ​ເປັນ​ເຫດການ​ທີ່ເກິດຂື້ນ​ໄດ້​ເປັນ​ປົກກະຕິ ໂດຍ​ສະເພາະ​ຢ່າງ​ຍິ່ງ​ໂປຣ​ເຈັກ​ໃຫຍ່​ໆ​ທີ່​ມີ Developer ຫຼາຍ​ໆ​ຄົນ​ຊ່ວຍ​ກັນ​​ຂຽນ​ໂຄ້​ດ ດັ່ງ​ນັ້ນ​, ການ​ແກ້ Conflict ຈຶ່ງ​ເປັນ​ໜຶ່ງ​ໃນ​ພື້ນ​ຖານ​ຂອງ​ການ​ໃຊ້​ງານ Git ທີ່​ນັກ​ພັດທະນາ​ຕ້ອງ​ເຂົ້າໃຈ​ ແລະ​ ຈັດການ​ກັບ​ມັນ​ບໍ່ໄດ້ ​ສະ​ນັ້ນ ​ຈະ​ເກີດ​ບັນຫາ​ຢ່າງ ​ເຊັ່ນ ຫຼົງໄປ​ລົບ​ໂຄ້​ດ​ຂອງ​ໝູ່ ໂດຍ​ບໍ່​ສົນ​ໃຈ​ຫຍັງເພື່ອ​ໃຫ້ Conflict ຫາຍ​ໄປເປັນ​ຕົ້ນ

* ເມື່ອ​ສະຫລຸບ​ຂັ້ນ​ຕອນ​ຂອງ​ການ​ແກ້ Conflict ຈະ​ເປັນ​ແບບ​ນີ້
* Pull ຈາກ Remote ລົງ​ເຄື່ອງ
* Conflict ເກີດ​ຂຶ້ນ
* ແກ້​ໄຂ​ໂຄ້​ດ​ໃຫ້​ເໝາະ​ສົມ
* Merge Commit
* Push ຂຶ້ນ Remote

## ສະຫລຸບ​ຮອບວຽຮ​ຂອງ Git ໃນ​ຊຸດທຳອິດ

* ເວລາ​ພັດທະນາ​ໂປຣ​ແກຣມກໍ​ຈະ​ມີ​ຂັ້ນ​ຕອນ​ແບບ​ນີ້
* ຂຽນ​ໂຄ້​ດ
* ເລືອກ​ໄຟ​ລ​ທີ່​ຕ້ອງ​ການ​ເຂົ້າ Staged
* Commit
* ຂຽນ​ໂຄ້​ດ​ຕໍ່
* ເລືອກ​ໄຟ​ລ​​ທີ່​ຕ້ອງ​ການ​ເຂົ້າ Staged
* Commit
* ວົນຫ​ລູບ​ໄປ​ເລື້ອຍ​ໆ​ຈົນ​ກວ່າ​ງານ​ຈະ​ສຳເລັດ
* ເວລາ​ຈະ Sync ຂໍ້​ມູນ​ໄປ​ທີ່ Remote ກໍ​ຈະ​ມີ​ຂັ້ນ​ຕອນ​ແບບ​ນີ້
* Fetch ເພື່ອ​ເຊັກ​ວ່າ​ມີ​ໃຜ Push ຫຍັງ​ຂຶ້ນ Remote ຫລື​ ບໍ່ (ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເຮັດ​ກໍ​ໄດ້)
* ຖ້າ​ມີ​ກໍ​ຈະ​ຕ້ອງ Pull ໂຄ້​ດ​ກ່ອນແລ້ວ Merge Commit ໃຫ້​ຮຽບຮ້ອຍ
* Push ຂໍ້​ມູນ​ຂຶ້ນ Remote
* ເວລາ​ຈະ Sync ຂໍ້​ມູນ​ໄປ​ທີ່ Remote ແບບ​ມີ Conflict ກໍ​ຈະ​ມີ​ຂັ້ນ​ຕອນ​ແບບ​ນີ້
* Fetch ເພື່ອ​ເຊັກ​ວ່າ​ມີ​ໃຜ Push ຫຍັງ​ຂຶ້ນ Remote ຫລື​ ບໍ່ (ບໍ່​ຈຳ​ເປັນ​ຕ້ອງ​ເຮັດ​ກໍ​ໄດ້)
* Pull ເພື່ອ​ດຶງ​ຂໍ້​ມູນ​ມາ​ໄວ້​ໃນ​ເຄື່ອງ
* ເກີດ Conflict
* ແກ້​ໄຂ​ໂຄ້​ດ​ໃຫ້​ເໝາະ​ສົມ
* Merge Commit
* Push ຂໍ້​ມູນ​ຂຶ້ນ Remote

# ຂຽນ​ໂຄ້​ດ​ໄປ​ຈົນ​ເຖິງ​ເມື່ອໃດ​ເຖິງ​ຈະ​ຕ້ອງ Commit?

ອັນ​ນີ້​ໜ້າ​ຈະ​ເປັນ​ຄຳ​ຖາມ​ທີ່ເກິດຂື້ນ​ເລື້ອຍ​ກັບ​ນັກ​ພັດທະນາ​ມື​ໃໝ່​ທີ່​ກຳ​ລັງ​ເລີ່ມ​ຕົ້ນ​ຮຽນ​ຮູ້​ ແລະ ​ຫັດ​ໃຊ້​ງານ Git ກັບ​ໂປຣ​ເຈັກ​ຂອງ​ຕົວ​ເອງ.

ໃນ​ການ Commit ແຕ່​ລະ​ເທື່ອ​ເຖິງ​ແມ່ນ​ວ່າ​ຈະ​ບໍ່​ໄດ້​ມີ​ໃຜ​ມາ​ກຳນົດ​ມາດຕະຖານ ແຕ່​ການ Commit ທີ່​ດີ​ຄວນ​ເຮັດໃຫ້​ສາມາດ​ໄລ່​ໂຄ້​ດ​ພາຍ​ຫລັງ​ໄດ້​ງ່າຍ ສາມາດ​ເຮັດວຽກ​ຮ່ວມ​ກັບ​ຄົນ​ອື່ນ​ໄດ້ ແລະ​ຕ້ອງ​ເຮັດ​ຄືເກົ່າ​ທຸກ​ເທື່ອ​ຈົນ​ເປັນ​ນິ​ໄສ ບໍ່​ແມ່ນ​ວ່າຄິດ​ອອກ​ເມື່ອໃດ​ກໍ​ຄ່ອຍ Commit.

ສຳລັບການ Commit ທີ່​ດີ​ຕ້ອງ​ກະ​ຊັບ ​ແລະ ​ນ້ອຍ​ຫລາຍ​ພໍ​ທີ່​ຈະ​ໄລ່​ເບິ່ງ​ໃນ History ໄດ້​ງ່າຍໆ ເວລາ​ມີ​ໂປຣ​ເຈັກ​​ໜຶ່ງ ຈະ​ຕ້ອງແຍກ​ສິ່ງ​ທີ່​ຕ້ອງ​ເຮັດ​​ອອກ​ມາ​ເປັນ Feature ແລ້ວ​ກໍ​ແບ່ງ​ຍ່ອຍ​ລົງ​ໄປ​ອີກ​ໃຫ້ເປັນ​ລະ​ດັບ Function ຈຶ່ງຈະ​ເລີ່ມ​ຂຽນ​ໂຄ້​ດ.

ເມື່ອ​ເຮັດ​ໃນ​ແຕ່​ລະ Function ສຳເລັດ​ກໍ​ຈະ Commit ໃຫ້​ຮຽບຮ້ອຍແລ້ວ​ກໍ​ເຮັດ Function ຕໍ່ໄປ ແລະ​ ເຮັດວົນ​ແບບ​ນີ້​ໄປ​ເລື້ອຍ​ໆຈົນ Feature ນັ້ນ​ສຳເລັດ.

ບໍ່​ແນະ​ນຳ​ໃຫ້​ເຮັດ​ໃນ​ລະ​ດັບ Feature ສຳເລັດ​ແລ້ວ​ຄ່ອຍ Commit ເພະ​ການເຮັດ​ແບບ​ນັ້ນ​​ຈະ​ໃຊ້​ປະໂຫຍດ Git ໄດ້​ແຕ່​ການ Backup ເທົ່າ​ນັ້ນ ແຕ່​ຈະ​ໄລ່​ເບິ່ງ History ເວລາ​ແກ້​ໄຂ​ໂຄ້​ດ​ໄດ້​ຍາກ ແລະ​ບໍ່​ສາມາດ​ຍ້ອນ​ຂໍ້​ມູນ​ກັບ​ໄດ້​ເລຍ ເພາະ​ມັນ​ຈະ​ຍ້ອນ​ກັບ​ໄປ​ທັງ​ໝົດ​ເລຍ ໂຄ້​ດ​ທີ່​ຂຽນ​ໄວ້​ທັງ​ໝົດ​ກໍ​ຈະ​ຫາຍ​ໄປ​ພ້ອມ​ໆ​ກັນ

ແລະ ​ກໍ​ບໍ່​ແນະ​ນຳ​ໃຫ້ Commit ໂຄ້​ດ​ທີ່​ຍັງ​ຂຽນ​ບໍ່​ສຳເລັດ​ເຊັ່ນ​ກັນ ໂຄ້​ດ​ທີ່​ຂຽນ​ບໍ່​ສຳເລັດ ​ໝາຍ​ເຖິງ​ໂຄ້​ດ​ທີ່​ບໍ່​ສາມາດ​ກົດ Run ເພື່ອ​ທົດສອບ​ໂປຣ​ແກຣມ​ໄດ້ ເພາະ​ຖ້າ​ເຮັດວຽກ​ຮ່ວມ​ກັບ​ຄົນ​ອື່ນ ແລ້ວ​ຄົນ​ອື່ນ Pull ຂໍ້​ມູນ​ໄປ ກາຍ​ເປັນ​ວ່າ​ຄົນ​ໆ​ນັ້ນ​ກໍ Run ໂປຣ​ແກຣມ​ບໍ່​ໄດ້​ເຊັ່ນ​ກັນ ຕ້ອງ​ເສຍ​ເວລາ​ນັ່ງ​ແກ້​ໂຄ້​ດ​ຂອງ​ເຮົາ​ອີກ ແລະ​ ຢ່າ​ລືມ​ໃສ່​ໃຈ​ກັບ Commit Message ຫ້າມໃສ່​ຫຍັງ​ແບບ​ນີ້​ໂດຍເດັດ​ຂາດ ເພາະ​ມັນ​ຈະ​ບໍ່​ມີ​ທາງ​ຮູ້​ເລຍ​ວ່າ Commit ນັ້ນ​ເຮັດ​ຫຍັງ​ລົງ​ໄປ

* “Edit code”
* “Commit”
* “Refactor”
* ຕ້ອງ Push ເມື່ອໃດ?

ນອກ​ເໜືອ​ຈາກ​ການ Commit ແລ້ວການ Push ກໍ​ເປັນ​ອີກ​ຢ່າງ​ທີ່​ຫຼາຍ​ໆ​ຄົນ​ສົງ​ໄສ​ເຊັ່ນ​ກັນ

ເນື່ອງ​ຈາກ​ການ Push ນັ້ນໃນ​ບາງເທື່ອ​ກໍ​ຕ້ອງ Pull ແລ້ວ Merge Commit ໃຫ້​ຮຽບຮ້ອຍ​ກ່ອນ​ເຖິງ​ຈະ​ເຮັດ​ໄດ້ ດັ່ງ​ນັ້ນ​ບໍ່​ແນະ​ນຳ​ໃຫ້ Push ກ່ອນ​ປິດ​ຄອມ​ ເນື່ອງ​ຈາກ​ຖ້າ​ມີ​ຄົນ Push ກ່ອນ​ໜ້າ ແລ້ວ​ພໍ​ເຈົ້າ Pull ມາ​ກໍ​ພົ​ບວ່າ​ມັນ Conflict ສະນັ້ນ ສຸດ​ທ້າຍ​ແລ້ວ​ເຈົ້າ​ກໍ​ຕ້ອງ​ນັ່ງ​ແກ້ Conflict ຊຸດ​ໃຫຍ່​ ດັ່ງ​ນັ້ນ​ທາງ​ທີ່​ດີ​ຄວນ Push ທຸກ​ເທື່ອ​ທີ່​ມີ​ໂອ​ກາດ​ດີກວ່າ ຢ່າງ​ນ້ອຍ​ຖ້າ​ມັນ​ຈະ Conflict ເຈົ້າ​ກໍ​ໄດ້​ແກ້​ຕັ້ງ​ແຕ່​ເນີ້ນ​ໆ ​ແລະ ​ບໍ່​ຫລາຍປານໃດ ທີ່​ສຳຄັນ​ມັນ​ຈະ​ຊ່ວຍ​ໃຫ້​ທຸກ​ຄົນ​ໃນ​ທີມ​ຮູ້ Progress ນຳກັນ​ໄດ້​ຈາກ​ສ​ະຖາ​ນະ​ການ Push ອີກດ້ວຍ ​ແລະ ​ນອກ​ຈາກ​ນີ້ ໃນ​ບາງ​ບາອນ​ທີ່​ມີ​ການເຮັດ Continuous Integration (CI) ທີ່​ຈະ​ດຶງ Commit ຈາກ Remote ໄປ​ເຮັດ Testing ຖ້າ​ເຮົາ Push ເລື້ອຍໆ​ແລ້ວ Test ເກີດ Failed ຂຶ້ນ​ມາ​ກໍ​ຈະ​ເຮັດໃຫ້​ຮູ້​ຕົວ​ໄດ້​ໄວ ​ແລະ​ ແກ້​ໄຂ​ໄດ້​ທັນ​ທີ **(ມາ​ຮຽນ​ຮູ້ Git ແບບ​ງ່າຍໆ​ກັນ​ເຖາະ** [**https://blog.nextzy.me/มาเรียนรู้-git-แบบง่ายๆกันเถอะ-427398e62f82**](https://blog.nextzy.me/มาเรียนรู้-git-แบบง่ายๆกันเถอะ-427398e62f82) **16 Aug 2017​).**

* + - 1. GraphQL API

GraphQL ຄືພາສາ​ສຳລັບ​ການ​ເຂົ້າ​ເຖິງ​ຂໍ້​ມູນ (Query Language) ເພື່ອ​ການ​ໃຊ້​ງານ API ຂອງ​ລະບົບ ​ແລະ ​ຈະ​ປະ​ມວນ​ຜົນ​ຄຳ​ສັ່ງ​ທີ່​ຝັ່ງ server ຫລື ​ທີ່​ເອີ້ນວ່າ server-side runtime ໂດຍ​ໃຊ້​ໂຄງ​ສ້າງ​ຂໍ້​ມູນ​ທີ່​ເຮົາ​ກຳນົດ​ໄວ້. ແຕ່ GraphQL ບໍ່​ໄດ້​ພັດທະນາ​ຂຶ້ນ​ມາ​ເພື່ອ​ແທນ​ທີ່​ພາສາ​ສຳລັບ​ການ​ເຂົ້າ​ເຖິງ​ຂໍ້​ມູນ​ເຊັ່ນ SQL ຫລື ​ເຮັດ​ຫນ້າທີ່​ເປັນ​ລະບົບ​ຈັດ​ເກັບ​ຖານ​ຂໍ້​ມູນ (storage engine) ແຕ່​ຢ່າງ​ໃດ

ໃນ​ປະຈຸບັນ​ວິທີ​ທີ່​ເຂົ້າ​ເຖິງ​ຂໍ້​ມູນ​ຜ່ານ​ເວັບ​ເຊີ​ວິດ​ທີ່​ໄດ້ຮັບ​ຄວາມ​ນິຍົມ​ກໍ​ຈະ​ເປັນ REST API ຜ່ານ​ທາງ HTTP Mehtods ຕັວ​ຢ່າງ​ເຊັ່ນ ຕ້ອງ​ການ​ດຶງ​ຂໍ້​ມູນ​ຜູ້​ໃຊ້​ງານ​ທັງ​ໝົດ ກໍ​ສາມາດ​ຮຽກ​ຜ່ານ API ໄດ້​ດັ່ງ​ນີ້

GET /users

ຫຼື ຫາກ​ຕ້ອງ​ການ​ດຶງ​ເພື່ອນ​ທັງ​ໝົດ​ຂອງ​ຜູ້​ໃຊ້​ງານ ID 25

GET /users/25/friends

ແຕ່​ຫາກ​​ຕ້ອງ​ການ​ດຶງ​ເບີໂທ​ສັບ​ຂອງ​ເພື່ອນ​ທີ່​ເປັນ​ເພື່ອນ​ກັບ​ຜູ້​ໃຊ້​ງານ ID 25 ຈະ​ຕ້ອງ​ດຶງ ​ແລະ ​ຂຽນ​ໂປຣ​ແກຣມ​ແບບໃດ? ໂດຍ​ຜົມ​ມີ​ການ​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ດັ່ງ​ນີ້

ຈາກ​ຕົວ​ຢ່າງ​ນີ້​ຈະ​ເຫັນ​ໄດ້​ວ່າ​ເຮົາ​ຈຳ​ເປັນ​ຕ້ອງ​ມີ API 2 end point ນຳກັນ ຄື GET /users/:id/friends ເພື່ອ​ດຶງ​ຂໍ້​ມູນ​ເພື່ອນ​ຂອງ​ຜູ້​ໃຊ້ ID 25 ຈາກ​ນັ້ນ​ກໍ​ວົນ​ລູບ​ດຶງ​ຂໍ້​ມູນ​ຜູ້​ໃຊ້​ໂດຍ​ໃຊ້ GET /users/:id ເພື່ອ​ດຶງ​ຂໍ້​ມູນ​ເບີໂທສັບ​ອີກຄັ້ງ ແລະ ​ວິທີ​ນີ້​ຜົມ​ກໍ​ຈະ​ໄດ້​ຂໍ້​ມູນ​ອື່ນ​ໆ​ທີ່​ບໍ່​ຈຳ​ເປັນ​ຂອງ​ຜູ້​ໃຊ້​ມາ​ອີກ​ດ້ວຍ ບໍ່​ວ່າ​ຈະ​ເປັນ​ຊື່ ຮູບ​ພາບ ແລະ ​ອີ​ເມ​ວ

ໃນ​ຄວາມ​ເປັນ​ຈິງ​ແລ້ວ ອົງ​ກອນໃຫຍ່​ໆທີ່​ມີ​ຫຼາກ​ຫຼາຍ​ພະແນກ​ ຕ່າງ​ກໍ​ຮ້ອງ​ຂໍ​ຂໍ້​ມູນ​ທີ່​ບໍ່​ຄືກັນ ເຊັ່ນ ບາງ​ພະແນກຕ້ອງ​ການ​ສະເພາະ​ຂໍ້​ມູ​ນພະ​ນັກ​ງານ ບາງ​ພະແນກຕ້ອງ​ການ​ຂໍ້​ມູ​ນພະນັກ​ງານ​ພ້ອມ​ຂໍ້​ມູນ​ເງິນ​ເດືອນ ຫລື​ ບາງ​ພະແນກ​ຕ້ອງ​ການ​ຂໍ້​ມູ​ນພະ​ນັກ​ງານ​ກັບ​ຈຳນວນ​ວັນທີ​ຂາດ​ ຫຼື ​ລາ​ເທົ່າ​ນັ້ນ ເປັນ​ຕົ້ນ ການ​ໃຊ້​ງານ REST API ຈຶ່ງ​ສ້າງ​ຄວາມ​ລຳ​ບາກ​ຕໍ່​ການ​ພັດທະນາ ​ແລະ ​ດູແລຮັກສາ​ເຊີ​ວິດ

ດ້ວຍ​ເຫດ​ນີ້ GraphQL ຈຶ່ງ​ໄດ້​ເຂົ້າ​ມາ​ເພື່ອ​ແກ້​ບັນຫາ​ໃນ​ຈຸດ​ນີ້​ໂດຍ​ສະເພາະ ເພື່ອ​ໃຫ້​ເຮົາ​ສາມາດ​ດຶງ​ຂໍ້​ມູນ​ໄດ້​ກົງ​ກັບ​ຄວາມ​ຕ້ອງ​ການ ລຸດ​ຄວາມ​ຊັບ​ຊ້ອນ​ໃນ​ການ​ຂຽນ​ໂຄ້​ດ​ເພື່ອ​ດຶງ​ຂໍ້​ມູນ ສາມາດ​ຈັດການ​ແລະ ​ດູແລຮັກສາ​ໂຄ້​ດ​ໃນ​ຝັ່ງ server-side ໄດ້​ງ່າຍ​ຫລາຍ​ຢິ່ງ​ຂຶ້ນ ແລະ ​ທີ່​ສຳຄັນ GraphQL ບໍ່​ໄດ້​ຜູກ​ຕິດ​ກັບ database ແລະ ທີ່​ຈັດ​ເກັບ​ຂໍ້​ມູນ​ໃດ​ໆ​ທັງ​ນັ້ນ **(ທຳຄວາມຮູ້ຈັກກັບ GraphQL** [**https://medium.com/@athivvat/ทำความรู้จักกับ-graphql-980ac00d6c17#:~:text=GraphQL%20คือ%20ภาษาสำหรับการเข้าถึงข้อมูล%20(Query%20Language)%20เพื่อ,ข้อมูลที่เรากำหนดไว้**](https://medium.com/@athivvat/ทำความรู้จักกับ-graphql-980ac00d6c17#:~:text=GraphQL%20คือ%20ภาษาสำหรับการเข้าถึงข้อมูล%20(Query%20Language)%20เพื่อ,ข้อมูลที่เรากำหนดไว้) **23 Feb 2020​).**

ວັນທີ ……/……./……… ວັນທີ ……/……./……… ວັນທີ ……/……./………

ລາຍເຊັນຄະນະກໍາມະການ ລາຍເຊັນອາຈານທີ່ປຶກສາ ລາຍເຊັນນັກສຶກສາ