شرحنا ثلاث أنواع من الموديلي بيو كونترولر، بعدين عملوا فيرجن حديث منه للويب، وبعدها بسموها موديلي بيو كونترولر ويب، single-page-application، js، وبعدها انتقلنا لأحدث إشي اللي هو الـ processingأو الـ business logicهدول بيستخدموا الـ client-side، server-side أو بصفحة البراوزر، بيكون عندك client-sideبصفحة الـ spring-java، الـ spring-boot، الـ البرمجي effortالثاني للموديلي بيو كونترولر، اللي اسمه موديلي بيو كونترولر ويب. وشفنا أنه الـ styleهدول كلهم بيشتغلوا على الـ عليه. هذا الحكي كله complexity اللي فيها web-applicationsبيعمله أصعب من التطبيق نفسه، وصمم للـ developerاللي الـ البسيطة applicationاللي بتحطه بالـ effortبس عشان تعرف أنه الـ exampleكامل، هذا الـ exampleحكينا ويمكن شرحنا الـ طبعاً حكينا Shutterstock ،البسيطة applicationsفي الموديلي بيو كونترولر، رح يكون غير مناسب أنك تستخدم له هاي الـ تجميع أو بتنحط في aggregation كل أجزاء التطبيق العملي أو كل الموديولز اللي بتبنيها، بنعملها monolithic-applicationالـ يعني لما تشغل التطبيق العملي جميع أجزاء التطبيق اللي ،run-modeطبعاً في الـ operating-systemبروسس واحدة على الـ على بروسس run-timeيعني هاد التطبيق كامل بيشتغل في الـ ،encapsulation هما عبارة عن موديولز بيكونوا، كلها بنعملها عالي بين الموديولز اللي موجودة عنه، فيه عنا خيارين dependency واحدة في نظامك التشغيل. بمعنى آخر إنه هاد التطبيق فيه إللي هي تزيد عدد الأجهزة اللي موجودة عنا. لكن بالنسبة لمشاكل المونولوتو تيك horizontal scalingحكينا، وفيه خيار تاني ال ليش؟ لأنه ببساطة هاد التطبيق اللي بيشتغلوا كل الأجزاء الموديولز ،scalabilityأبليكيشن، بيعاني من مشكلة من ناحية ال الموجودة فيه في بروسس واحدة، متطرة أجيب جهاز جديد وأنقل كل الموديولز الموجودة بالتطبيق على الجهاز الجديد، زاد عدد لكل التطبيق العملي اللي scaling لموديول 6 في المونولوتو تيك أبليكيشن، لازم أعمل scaling المستخدمين عليه، ما بقدر أعمل زي سيرفرات، عشان بس تعمل resources ليه؟ لأنك إنت عم تستهلك ،efficient موجود على جهاز آخر، وهذا الحكي مش لسيرفرات معينة من هذا التطبيق الموجود، فهذا تعتبر من المشاكل الأساسية الموجودة في المونولوتو تيك أبليكيشن. scaling اللي هي تحرير نسخ من هذا الابليكيشن بطيئة جداً، ليه؟ لأنهم التيم اللي بيشتغلوا على هذا الابليكيشن، ما عندهم يعني بمعنى آخر السبب إنه أي تغيير في أي موديول من هاي الموديولز، لازم يكون جميع الموديولز عارفين عنها. ببساطة اللي ،permission_ال بيشتغلوا في الابليكيشن، وباقي الموديول جاهزين، لهاي الموديولز الجاهزة، لحد ما developer بيصير كالتالي، ممكن يكون عدة لهم releases لكل الموديولز مع بعضها، وبعدين ينعمل testing وبعدين ينعمل ،development_ينتهوا شميع الموديولز من ال موجود micro service_موجودة في بروسس واحدة، ك monolithic application_مرة واحدة، كل الموديولز الموجودة في ال micro_زي ما حكينا من الموديولز، كل شق منهم يعتبر ،micro_على بروسس، هذا الموديولز موجود على بروسس، وكل رقم 5، module ليه؟ لأنه ببساطة، اللي فيها .self-contained بحد ذاتها، يعني هم بيحكوا مع بعض، لكن هم بشكل عام service الأخرى، services_عن ال independent لحالها، أو يعني ضغط من قبل اليوزر، لأنه service_لهاي ال installation أو عملية أسهل، ليه؟ monolithic_مقارنة بال ،application_لكل ال installation وتنقله على سيرفر آخر، مش بالضرورة تعمل عملية installation، بيقدر يعملها developer_بتشتغل بشكل منفصل، طبعاً، فمباشرة ال process كل ،process لأنه ببساطة كل يعني على سيرفرات رئيسية وتكون ،production side_اللي موجودة، على ال micro service_من هاي ال release ويعمل على micro service_إنه ممكن يتبرمج كل ،monolithic_تنتظر بعضها. مقارنة بال micro services_شغالة، أو كل ال فيهم. والشغلة كمان multi-technologies فممكن تدعم ،angular JS_تكنولوجيا معينة، وأبرمجها على جزء منها على ال بحد application كل واحدة كأنها services_المهمة إنه الفيليارز اللي بيصير في التطبيق العملي بارشالي، لأنهم أصلاً هدول ال من بعض، فممكن يعمل لك requests بيطلبوا .micro service architecture_ذاتها، فهذا الإشي جاي مميز فعلياً بال فبيبطئ لك التطبيق العملي بشكل عام، ،network عن congestion أو network عن latency عالي أو communication _أو عدة devices_على عدة micro services distributed_والمشكلة الثالثة الموجودة عنا إنه ممكن يكون لأنها هاي ال يعني بتضل ،preserve فيه خلل، بتضل ،queuing اللي هو المسج ،servicesـبين هاي ال data_فعملية نقل ال servers، وتوزحها عدد سيرفرات موجودة ،priority queue على سيرفر واحد، كيف لي ،scalable النظام ،queueمحفوظة عندك بال ما client؟ يعني الasynchronous شو يعني ،asynchronous، design، هلا، الاشي المهم كمان اللي موجود بهذا ال clientبراحته، بين ال processing والسيرفر بيعملها عملية ،queueبيهتم، بيرمد المسجات، المسجات كلها بتتخزن بال time والسيرفر يشتغلوا، كل واحد بيشتغل بشكل منفصل، وهذا بيعمل لك كمان clientوالسيرفر، هذا بيسمح لك إنه ال

message ببساطة، حتى لو صار فيه اختلاف في البودرات، ففيه كتير تطبيقات بتشتغل، بتنبنى على شكل decoupling، الآن شوف فكرته ببساطة، فكرته ببساطة إنه، لازم يكون .designببساطة، لهذا ال observer patternاللي هو ال queuing، له. بس هون بشكل عام بيحكي لك المسجات، extension تمام، أو listenersاتمام، لازم يكون هدول ال publisher عندي أو بالنظام، اللي subscriber، أو listener إنه message queuing in addition، حرفياً design فهو نفس event بيسموها already، بيكونوا مسجلين eventsمعين، لمجموعة من ال Jevents، observerأو ال notificationلل receive هو بيعمل بيكونوا عاملين notification، أو لخلينا نحكي اللي بيعمل observerفببساطة هو بيشبه نظام ال ublisherعند مين؟ عند الب اللي موجودين subsystemsلل notification مباشرة بنعمل ublisherبيصير عند الب event عندهم، وأي registration عشان إذا صار فيه اختلاف البودرات بين السيستيم الأول، بتتخزن queue، بتخزنوا داخل eventsعندهم، طبعاً هدول ال الآن pipe and filter architecture، إلو. هي اسمه extension الأول، أو بمعنى آخر designالمعلومات كلها، فهو شبيه بال المطلوب منك عشان تكون عارف كيف طبيعة الماتريال هاي، على الأغلب بجيب لك نظام، طبعاً؟ ممكن أجي أحكي لك عندي برسم، description بكون حكيت لك، notification بدو يعمل sales system، sales system، وعندك airline system، event، لل publish هو اللي بعمل publisherأن ال publisher subscriber، معين، على شكل على سبيل المثال تبعه كيف بيشتغل، ما abstract levelالكورسبوند إلو، ال diagramوترسم لل notification، هو اللي بستقبل subscriberوال تمام؟ في pipes، مع بعضها عن طريق ال communication أكيد تعرف كل واحد شو الفوائد، تمام؟ ونتيجة هذا الفلتر، بتعمل أشهر التطبيقات العملية اليونيكس كومند، احنا لما نكتب على اليونيكس كومند designكتير تطبيقات عملية بنيت على هذا ال اللي داخلها، بتفلترها حسب. هذا فلوت، contentsمعين، عن طريق الكومند لاين، تمام؟ بتم ترجمته، وبعدين ترتبها، أو ترتب ال فتدخل على موضوع السيمانتيك أناليسيز، كل بلوك من هذه البلوكات أو كل فلتر عبارة عن برنامج كامل متكامل برمجياً، ستبرمج لغزيكال أناليسيز، برنامج يعمل التوكينيزيشن، برنامج يعمل السينتاكس أناليسيز، ستفهم برمجياً كيف فعلياً الابليكيشن اللي بتكتب وبنعمله كومبيليشن بشكل عام بأي لغة برمجي، بتطلع منها انستركشن جديد، اللي هي تطلع الانترمديت لانجويج، الانترمديت لانجويج اللي هي أسيمبلي لانجويج، قبل ما تحولها لماشين لانجويج اللي موجودة علينا، طبعاً فدول كل بلوك منهم عبارة عن أبليكيشن كامل متكامل، كتير كتير كبير بس أنا بحكي، ممكن تعمل كومبايلر بسيط مكون من كم جملة برمجية طبعاً، لكن بشكل عام الكومبايلرات بتكون فيها كومبليكسيتي عالية في بناء. يعني ما راح بكورس تبني كومبايلر لانسي ولا جافا، ما راح تبني لبرنامج بسيط دوميني سبيسيفيك لانجويج. اللي مطلوب منك انك تعرف انه هاي بعض التطبيقات العملية اللي بتستخدم مبدأ البيب اند فلتر. شو فيه أخرى تطبيقات ممكن تستخدم مبدأ البيب اند فلتر عندك الانتر برايس البزنس، ويركي فلو، ويركي فلو كيف بدك تدفع فاتورة معينة، بتأول إشي بتتم إصدار الفاتورة تقرأها، تحدد قديش مقدار البيمنت، يا إما بتشوف امتى آخر موعد لتدفع، وبعدين بتحط ريمايندر انك رح تدفع باليوم لفلاني، أو مباشرة بتدفع تاخد وصل نتيجة انك دفعت، يعني أمامك خيارين يا بتدفع بشكل مباشر يا بتأجل الدفع بمعنى أخر، كل بلوك فيهم فلتر بعملك أكشن معين، هلا بشكل عام، شو بتشوف فوائد ولميتيشن في هذا الديزاين؟ أول شغلة، أو في حدا عامل كومبايلر، وأكيد الليجزيكال أناليسيز ما رح يختلف، وين ما تحطه بأي لغة برمجينة، رح يكون البرنامج اللي بيعمل توكينيزيشن نفسه، يعني منفصل لهاي الفلترات عن بعضها، بس بتستخدم السيربيسز من بعضها، ففيري إيزي تعمل ريوز للابليكيشن. كتير مناسب بالويركي فلو، يعني أي اشي بتلاقي فيه ويركي فلو، هذا فيك ببساطة تروح تبنيه على شكل بايب أنت فلتر. لأي فلتر سهل جداً، انك تعمل وتضيف تعمله انتجريشن بالسيستم، ممكن يشتغل سيكوانشالي وممكن what the، يشتغل كونكرانتلي، هذا سيكوانشالي وراء بعض بس هذا كونكرانتلي، لأنه انا ممكن اعمل فورك امشي بهذا الفلو امشي بهذا الفلو، فبشتغل على كل مبادئ الفوركينج ديزاين، ليه ببساطة سيكوانشالي او كونكرانتلي. هلا وين same moment عالسيستم، طبعاً يعني انت تخيل هذا overheadالمشكلة في هذا الديزاين؟ المشكلة انه اذا زادت عدد البلوكات او الفلاتر، بزيد ال النظام، ويبعطه للي بعده انت، هاي مشكلة من المشاكل الموجودة. المشكلة التاني في الديزاين انه، اذا كان الفلتر الاول مش على json اللي بيطلع من هذا البلوك عبارة عن output رح يصير في عنا خلال في النظام ليه؟ لانه تخيل ال data format، علم بال طريقة وصف المعلومات data اذا اختلف الفورمات ال xml file واللي بيستقبلوا البلوك التاني او الفلتر التاني عبارة عن file، اللي بتنتقل من طرف الى طرف اخر، لازم يكون متفق عليه agreed، بين الاطراف بصير عنا خلال بالنظام. فلازم يكون الطرفين لاي مقطع منهم لازم اكون على reusing، فمعناها انه انا اذا بدي اعمل transformation، مسبقاً، واقل بصير عنا خلال في عملية

بتطلع لك ،design يعني تخيل معي هذا ال ،data اللي طلع فورمات ال data structure و نوع ال data structure علم بنوع ال linked وهذا بيستقبل لكها عشاكل ،tree فيه عشاكل data اللي طلع فورمات ال data structure نوع ال ،tree عشاكل data ليه لانه بتاخد .pipe and filter architecture design اللي ممكن الواحد يشوفها في ال limitation طبعاً هاي من اكتر ال ،list outliers، مرات بيكون فيها ،corruption data مرات بيكون فيها ،data تنظف ال ،csv file من excel sheet من data ال بعدين ،outliers تشيل data لل cleaning مثلاً تبها 40، فش درجة حرارة بتوصل 200 سلسيوس عصابين المدالة، فبدك تعمل معناها، في عندك عصابين feature engineering شو ،feature engineering او بيسموها ،data engineering بتعمل هلق فيه مجموعة من ،acilitiesمن الف close لل بيوت، يعتمد على كم غرفة موجود فيه، او يعني قديش هو price للمثال، ال هذي الاربعة والخمسة، بدي input على price based على سبيل المثال ال prediction المعلومات موجودة، وانا اللي بدي اعمله شو feature engineeringلسعر البيت، فهذا ال related اشوف سعر البيت هذي، بس فيه معلومة مش كتير مهمة او مش للفيتشارز اللي remove او اعمل reduction معناها انه بدي اشوف من هاي الفيتشارز اعمل eature engineeringمعناها، الف ممكن كون حجمها data للفيتشارز اللي مش مهمة، لانه اذا بدي استخدم كل ال remove او اعمل reduction مش مهمة، اعمل feature الموديل، فمنعمل اشي منسميه performance كلها، فبالتالي ممكن يتأثر على columnsضخم جداً او ال validation لترينج للموديل، وبعدين بتروح لعملية feature engineering بعدين بتروح النتيجة بعد ما تعمل ،engineering بتشتغل على مبدأ machine learning approach بنسميهم او اي style فهدول كلهم .deploymentالموديل، وبعدين بتروح لل اللي اخدوا ،beer to beer architectureعندنا اللي هو ال style نيجي لاخر Shutterstock، pipe and filter style اللي هو اللي بسموها versionبسموه، بس انا بحكي هلأ عن ال point to pointاو ال beer to beerنتويرك اكيد سمعوا بهذا اللي هو ال ،infrastructure-less المشة اللي معناها انت بتشبك الاجهزة بعض بعض بدون ما يكون عندك اي هيكلية داخلية ،mesh for monitoring ،تخيل هذا السيرفر designمعين. لكن ان وجد سيرفر في ال structure معين، ملهاش infrastructure ملهاش للapplication، اللي هو احنا كنا نسميه زمان ،semi-centralized architecture لكن هذا distributed architecture بالعادة فعليا، من الفوائد الكبيرة انه الاجهزة wireless networks بيحكوا بشكل مباشر مع بعض، يعني هذا كتير مستخدم في clientsال wirelessالموجود عند الاجهزة والاخرى في الشبكات. هدول ال storageوال computational power بتصير تستفيد من computation وبيدهم يعملوا ،environmentبيقرؤوا ال sensorsمزروعين في بيئة معينة، تمام وبيستقبلوا هدول ال sensors beer to beerهذا ال designالفكرة من ال ،low level الموجود. يعني خلصت البطارية او صارت في environmentعلى ال based الموجود عند storageللاجهزة الاخرى، عشان يستفيد من الباور وال computationلل transfer انه بيقدر يعمل عملية beer بشكل عام، الاستفادة beer to beerبين بعضها. فهاي فائدة ال computationالاجهزة الاخرى فبتصير الاجهزة تتنقل عملية ال hard disksال storageزي البطاريات ال computation powerالمقصود فيها ال resources الموجود resourcesمن ال وقد server من هاي قد تكون node كامل في الشبكة الموجودة. تمام كل computationعشان تعمل تكمل ال ،and so on يعني ،resourcesفي حدا بزودك بال ،consuming الموجودة في حدا بعمل resourcesوكل طبعا لاحظ ال ،clients تكون او يعني يطلب جهاز 13 من جهاز 12 يعمل power فممكن يعطي لهذا الجهاز computation ممكن هذا يكون جهاز ماعمل رقم 12، فبصيروا يستفيدوا من node رقم 12 على سبيل المثال ممكن هذا يخزن عند node عند computationال اللي هو bit torrentزي هيك كتير بيستخدم ال .applicationفي ال collaboration اللي موجودة عندنا وهذا بعمل resourcesال ليه لانه ببساطة انت لما تيجي تسايب على جهاز ممكن هذا الجهاز تكون file sharingفي ال downloadعملية ال ،file sharing لهاي الملفات sharing تخزن عليها وتعمل 8 node اللي عند storageاستهلكت، فممكن تستغل ال storage تبعته resources architecture styleهذا من اكتر ال ،networkingكتير مستخدم لل peer to peer architectureعليها، فهاي ببساطة مبدأ ال ببساطة راح تشتغل على networking بين الاجهزة، يعني اذا طبيعة التطبيق العامل يتبعك networkاللي راح تشوفه في موضوع ال هذا. هاي يا جماعة كل الاجهزة كل الأحوال مبلع يا علي مبلع، بس ما هو شو الفكرة كانوا عشان هيك كنت احكي لك عن عندهم بهدف حفظ السيكوريتي registration السيرفر الرئيسي اللي بيكون كل المستخدمين هدول عاملين semi-centralizedال اللي ممكن انت تستغلها لبناء انواع architecture stylesاللي موجود، لكن فيه بروتوكولات بتضبط هذا الحكي. هاي جميع ال وانطلاقا باي نوع ،webالمخصصين لل model view controllerوال layeredمن ال starting ،مختلفة من التطبيقات العملية

حكينا modelوال layeredيعني فيه كتير يعني هون ال ،networking system زي compiler اخر من التطبيقات العملية زي متخصص اكتر في التطبيقات peer to peerال ،scalability اذا بدك تبني التطبيق وتستغل فكرة web microservice المستخدمة جدا في بناء architecture styles very commonفهاي معظم ال .networking اللي فيها networkingال التطبيقات العملية up to date يعني لحد ايامنا هاي كلها بتستخدم في بناء التطبيقات العملية. بيركز على ال design principles الخاصة في ال propertiesال decouplingال couplingال ،solid principlesال .design يعني we assume اننا جمعنا وهلأ بنروح لمرحلة التستين. طبعا في كتير شغلات implementationعملنا ال designعملنا ال requirements، we assume عليها ان شاء الله لحق اخلصها خلال الفصل بس رح اروح هيلا جنب على الشغلات الاساسية عشان نضبط skip انا رح اعمل the الماتريال تكون اخدت كل الافكار الاساسية بالمدى طبعا. فهلأ رح نروح لموضوع اللي بيركز بشكل عام اللي بيركز على للكود اللي how to improve the quality للكود اللي انت كتبه، احنا بلشنا وشرحنا شق معين يعني انا هلأ اللي بهمني quality هذا موضوع جدا مهم test-driven development اليوم بدي احكي عن موضوع اخر بالتستنج اسمه Shutterstock ،عنده تقريبا سهل test-driven development رح يكون بفيز رقم 2 بالمشروع، خلينا نروح لموضوع static analysisوموضوع ال ببساطة كيف تكتب التست كيسز test-driven development جدا هذا الموضوع قريبا انت عامله بس بطريقة عكسية، مفهوم نفسها لتطبيق هذا المبدأ unit طبعا، احنا رح نرجع نستغل الجاي production codeقبل ما تبني اي سطر كود قبل ما تعمل ال اللي هو اسمه test-driven development.