***GOTOVE KOMPONENTE***

Korišćenje Gotovih stvari, elemenata i alata u našim rešenjima.

Prvi razlog zašto bi koristili gotove komponente je da ne moramo mi da razvijamo ali nije jedini razlog ali najčešće nije ni najvažniji ni presudan.

Komponente koje možemo koristiti u razvoju(kalse gotovih stvari koje nam olaksavaju razvoj):

1. **softverske bibilioteke (softwere library)**

* nešto što implementira neki skup funkcionalnosti što možemo da ugradimo u svoju aplikaciji (postaje deo naše aplikacije) i kada nam je potrebna ta funkcionalnost pozovemo biblioteku, ona odradi neki posao i vrati rezultat
* Primer: pretvaranje teksta u sliku (Image Magic) – jednostavniji zadatak za implementaciju ,pretvaranje slike u tekst(Abbyy, OCR algoritam) – kompikovanije rešavanje ,ako nam samo treba da pretvorimo iz slike u tekst i dalje radimo sa tim onda ne moramo to da implementiramo već možemo da radimo ono što nam je bitno

1. **radni okvir (framework)**

* nešto što implementira često korišćene mehanizme koji se javljaju u nekoj primeni ( koji nam svaki put trebaju pri razvoju određene vrste aplikacija)
* karakteristično je da je taj mehanizam implementiran da izvršava niz nekih koraka, ali sam po sebi ne daje nikakvu konkretnu funkcionalnost, već je u framework potrebno ugraditi svoje komponente koje će da odrade konkretnu stvar koja nama treba u određenom koraku i da svoj rezultat vrate radnom okviru
* on poziva nas i mi njemu vraćamo rezultat
* princip inverzije kontrole, holivud princip – kada glumci idu na kastinge, „*Dont call us, we’ll call you*“
* primer: spring
  + radni okvir za razvoj aplikacija (mobilnih, desktop aplikacija, konzolnih, gui,..) čime se mogu pokriti razne tehnologije
  + zbog širokog polja primene veoma je velik i bilo bi nezgodno da u svaku aplikaciju moramo da uključimo čitav spring i zato je organizovan u module, koje možemo po potrebi uključiti za onu vrstu aplikacije i tehnologije koje želimo da imamo u nekom konkretnom slučaju
* interakcija radnih okvira i biblioteka – biblioteke može da zove i sam radni okvir, ali i mi unutar naših komponenata koje ugrađujemo u radni okvir

1. **servis (service)**

* kada nam zatereba neka funkcionalnost pozovemo ga on uradi posao i vrati rezulatat
* razlika servisa i biblioteke - biblioteku ugrađujemo u aplikaciju, a servisi su nezavisni od nas, oni se zasebno izvršavaju
* servisi mogu biti:
  + lokalni - na istom računaru kao i aplikacija koja se izvršava i koja ih poziva
  + udaljenog tipa - na nekom drugom računaru
* web servisi – udaljeni servis kog pozivamo preko računarske mreže koja je dostupna na web-u
* primer:
  + lokalni – Windows Print Spooler
    - kada iz aplikacije želimo da štampamo ona to pošalje štampaču preko servisa operativnog sistem (spoole-a), nakon čega mi možemo da nastavimo da editujemo, snimam datoteku, zatvorim,.. a spooler brine o tome koji je to konkretno štampač, koja je veličina papira, da li ima papira, da li se nešto već štampa,..
  + web servis - Google Cloud Vision
    - rešava isti problem koji je rešavala biblioteka abbyy, ali ako imamo mrežu na raspolaganju možemo njemu da pošaljemo i on nam vrati string (dugi pristup istom problemu)
    - prednost:
      * istom servisu mogu pristupati razne aplikacije - za razvoj algoritama za ocr treba da imamo širok spektar ulaznih podataka (slika) i da znamo koliko je dobro ono što smo vratili kao razultat
      * korišćenjem web servisa omogućavamo tom servisu da poboljša svoju uslugu pružajući mu podatke koje im mi saljemo
    - mana: koriste naše podatke za unapređivanje funkcionalnosti
  + Primer : aplikacija za ocitavanje mehanickih elektricnih brojlia, usmeravanjem kamere telefona na brojcanik aplikacija posalje google-u koji uradi ocr i vrati cifre na telefon, samo što google zna stanje električnih brojila za dobar deo zemlje

1. **korisniske kontrole (UI control)**

* delovi grafičkog korisničkog interfejsa, omogućavaju da izmenimo ili prikažemo neki podatak (vizualizaciju podataka za prikaz grafikona) ili su namenjeni pokretanju neke funkcionalnosti (polja za unos, dugmici, meniji,..) sve to bi bilo nezgodnoza razvoj od nule
* primer: kalendar, imamo gotovu korisničku kontrolu koja se može pozivati iz svinga i relativno jednostavno možemo na neko mesto na grafičkom korisničkom interfejsu da postavimo odabir datuma
* međutim često ćemo hteti nešto da izmenimo u zavisnosti od kulture kojima su namenjeni kako bi im olakšali (oblik datuma,jezik,..) što nije jednostavno za implementaciju

1. **biblioteke korisničkih kontrola (UI Library)**

* skup korisničkih kontrola raznih namena koje imaju sličan izgled i na sličan način se koriste (**look&feel**)
* koristimo ih:
  + korisnički interfejsi (linux, windows, mac) izgledaju različito pa moramo prilagoditi izgled i način korišćenja ovih kontrola platformi gde se koristi – sving pravi g.intefejse za javu te je neophodno da ima mogućnost odabira look&feel
  + želimo da način na koji se koristi taj inrefejs bude uniforman unutar naše aplikacije - da se stvari na jednoj formi naše aplikacije koja predstavlja jednan entitet rad na isti način kao što se radi za neku drugi entitet (kada razvijaju početnici pa žele da pokažu svoje znanje, ili više loše koordinisanih programera)

1. **engine**

* definicija zavisi od konteksta:
  + kod razvoja igre - engin se bavi prikazivanjem grafike,prikupljenjem korisničkih akcija
  + nekima je engine isto što i framewor - implementira odredjene meh za izvršavanje aplikacije
  + može biti i radni okvir koji je toliko detaljno razvijen da za njega ne programiramo funkcionalnosti već mu samo damo specifikaciju kako naši entiteti izgledaju i koje funkcionalnosti su moguće nad njima i sam endin odradi ostalo (operacije, veze između entiteta,..)

1. **alati (tools)**

* ne ugrađujemo ih u aplikaciju i isporučujemo ih sa aplikacijom i najčešće ona ne poziva alat
* alate koristimo tokom razvoja i oni nam olakšavaju sam razvoj tako što automatizuju neke stvari koje se svaki put rade kada se razvija aplikacija
* primeri:
  + Maven
    - alat za igradnju aplikacije (build tool), on podrazumeva sve što je nepohodno uraditi da bi se aplikacija prevela iz nekog izvornog oblika u neki oblik koji se može pokrenuti/izvršavati
    - (pored kompajliranja, potrebno je dobaviti sve biblioteke, frameworke, sve prateće komponente koje smo koristili, pri čemu i one mogu zavisiti od nečega drugog što je potrebno takođe dobaviti)
    - jedna od stvari koje Meven rešava je upravljenj nezavisnostima - na osnovu naše specifikacije, sve od čega zavisi aplikacija skine sa interneta uzme njegovu specifikaciju i to dobavi i tako sve u krug dok se ne dobavi sve ili dok se ne utvrdi da ima neki problem u čitavom lancu zavisnosti, nakon tog može da pokrene automatizovane testova aplikacije, da je zapakuje u neku celinu,..
    - obavlja čitav niz aktivnosti koje se obavljaju automatizovano svakaki put kada build-ujemo aplikaciju
  + Git
    - sistem za kontrolu verzija
    - koriste se da bi pratili kako izvorni kod evaluira vremenom i da se omogući da više ljudi radi nad istim izvornim kodom, da se njihove izmene sinhronizuju
    - esencijalna stvar da bi bili efikasni u bilo kakvom timskom radu
    - ima i niz propratnih korisnih funkcionanosti koji omogućavaju da se utvrdi kada se neka izmena pojavila, tako utvrdi kada je u kod uneto nešto što je uzrok greške koja se kasnije ispoljila, koliko široke moraju biti izmene koje će otkloniti tu grešku, koliko će ta izmena uneti posledice u razne verzije koje su isporučene raznim grupama korisnika,..

***ODABIR GOTOVIH KOMPONENTI***

Kada smo odlučili da hoćemo da nađemo nešto gotovo kako da odaberemo komponentu koju ćemo koristiti:

1. **Da li postoji?**

* ispitivanjem i raspitivanjem moramo otkriti da li postoji nešto što rešava naš problem

1. **Da li je komponenta testirana?**

* ako postoji više kandidata, više komponenti koji impelementiraju funkcionalnosti
* da li možemo doći do testova koji su korištećni prilikom razvoja, ako jeste testirana na pravi način skratićemo vreme za razvoj svoje aplikacije
* ako nije testirana možda je bolje korisiti drugi ili sam razviti

1. **Da li je dokumentovana?**

* i sa tehničke i korisničke strane, ako ću morati da potrošim dugo vremena da bi shvatili kako da je upotrebimo mozda je lakse da je napravimo, ali ako ima daje neko poverenje da je dobro napravljena

1. **Da li je oprobana u praksi?**

* ukoliko imamo tuđa iskustva pozitivna ili negativna, ona pomažu u odabiru i u isto vreme ukoliko je to nešto što duže vreme postoji možemo očekivati da je neko već naišao na sve trivijalne probleme u koršćenju, uočio grekše i otklonio ih pa to nećemo morati mi

1. **Da li postoji podrska?**

* ako razvijamo softver za profesionalnu primenu od nas će se očekivati da damo podršku za naš softver a dobro bi bilo i da imamo i podršku za komponenete koje koristimo
* najcesce podrska za te komponente nije besplatna i ako je samama komponenta besplatna ali sa druge strane to je nešto što nam može pomoći da potencionalne probleme rešimo u roku koji se od nas očekuje

1. **Da li se isplati?**

* ako ne postoji besplatna varijanta, da li je jeftinije da kupim nego da razvijam sam
* uvek treba računati i koliko košta naše vreme da se nešto implementira
* nekada je isplativije da platimo kvalitetno rešenje nego da trošimo trud na prilagođavanje nekog manje kvalitetnog rešenja koje je besplatno
* ne znači da će uvek stvari koje se plaćaju biti kvalitetnije, često je i obrnuto ali ne treba odma odbaciti nešto što se plaća jer se iz početne perspektive čini skupo, neka da će uštedeti mnogo truda

1. **Da li smemo da je koristimo?**

* pitanje softverskih licenci i suština je da li licenca sa kojom dolazi komponenta je kompatibilna sa licencom sa kojom se isporučuje ono u šta ugrađujemo komponentu
  + da li je dozvoljeno da korisitmo komponentu u vrsti aplikacije koju razvijamo
  + da li ako koristimo neku komponentu moramo da menjamo svoju licencu tj. uslove korišćenja naše aplikacije
* u velikim kompanijama uglavnom postoje procedure u kojima se odobrava korišćenje tih komponenti u proizvodu koji se razvija, prvo se pravnoj službi prosledi upit i oni utvrđuju da li sme da se koristi ili ne

***PREDNSOTI I MANE***

|  |  |
| --- | --- |
| Ne moramo mi da implementiramo. | Ne znamo kako je implementirano, ne znamo detalje funkcionisanja, da li je to najbolji način, ako nešto treba menjati pa i da je komponenta otvorenog koda morali bi da utrošimo vreme da utvrdimo kako su stvari impelemnrtirane. |
| Štedimo vreme za razvoj. | Ne smemo zanemariti i vreme koje je nephodno da ovladamo nečim novim. |
| Testirane |  |
| Oprobane u praksi |  |
|  | Teško menjati iznutra, ali i ako ta komponenta ne prati određeni standar teško je i zameniti komponentu koju koristimo. |
|  | Zavisnost od onog koji je proizveo komponentu (ne želi više da razvija i pruža podršku, ne ispalti mu se, povuče je sa tržišta i ostajemo sa rupom u našem rešenju)  Postoje neki mehanizmi koje možemo primeniti u organizaciji sopstvene arhitekture tako da prelazak na alternativne komponente bude lakši ( možemo uneti neke svoje interne slojeve prilagođavanja koje ćemo u budućnosti koristi ukoliko treba promeniti neku komponetnu).  LOCK IN |
| **RAZVOJ SOPSTVENE GOTOVE KOMPONENTE** | |
| Razviti stvari koje se mogu koristiti ponovo jeste složenije međutim to je pametno uloženo vreme i napor, ukoliko vidimo da ćemo to korisiti i sutra i time olakšati neki budući razvoj. | Teže je praviti delove rešenja tako da oni budu nešto što može da se koristi gotovo (u drugim delovima istog projekta/aplikacije ili nekim budućim našim ili tuđim aplikacijama). |
| To je svakako dobra stvar i poželjno je, i znak kvalitetnog rešenja, ona će i nama koristiti kada posle određenog vremena budemo hteti da je korisitimo. | Moramo pisati dokumentaciju. |
| Svakako obavezno ako je to ozbiljan i profesionalan razvoj. | Moramo je testirati. |
| Možemo da ih učinimo dostupnim durgima | Nisu oprobane kao tuđe. |
| Možemo ih prodati. | Ceo razvoj košta. |
|  | Time što smo sami razvili komponentu ne rešavamo problem lock in-a.  Ako se čitav ostatak nekog problema vrti oko toga da poziva neku komponentu koja obavlja suštinski posao i veliki napor smo uložili u razvoj komponente neće biti jednostavan proces napuštanje te komponente (možemo jedino koristiti mehanizme kao i prošli put). |