PRIJEDLOZI, PRIMJERI I UPUTE ZA IMPLEMENTACIJU POJEDINIH FUNKCIONALNOSTI UNUTAR PROJEKTA

Navedeni prijedlozi i primjeri nisu obavezni. Štoviše, poželjno je da studenti budu što originalniji pri implementaciji pojedinih funkcionalnosti. Implementacija funkcionalnosti koje su u potpunosti iste (ili tek minimalno izmijenjene) kao one na predavanjima, u gotovim primjerima i videima neće biti bodovane. Zbog toga se studenti mole da sukladno uputama prije obrane projekta svom dodijeljenom nastavku pošalju opis funkcionalnosti u dokumentu koji se nalazi na LMS-2020 stranicama kolegija kako bi iste mogle biti pravovremeno provjerene i kako bi studenti bili sigurni da za njih mogu dobiti maksimalne bodove.

**1. Aplikacija sadrži minimalno dva nova korisnička tipa (klase) s minimalno po dva atributa i dvije metode (2 boda). U slučaju kreiranje i korištenja više klasa za ovu funkcionalnost moguće je dobiti 3 boda. U aplikaciji je potrebno demonstrirati upotrebu navedenih klasa, atributa i metoda.**

Ovisno o projektu koji razvijate možete se odlučiti na uvođenje novih tipova podataka (klasa). Npr.

1. Klasa koja sadrži određene podatke (atribute) o korisniku aplikacije te metode za prijavu i odjavu korisnika iz aplikacije.
2. Klasa koja sadrži podatke o akcijama tj. logovima aplikacije (tko je kada i što radio unutar aplikacije) te ima metode za dodavanje novog zapisa, brisanje postojećeg zapisa te dohvaćanje (vraćanje) zapisa iz nekog vremenskog perioda.
3. Klasa koja sadrži metode za spremanje logova u datoteke raznih formata (TXT, XML, JSON itd.) gdje se atributima klase određuje naziv i format odredišne datoteke.
4. Klasa koja sadrži atribute i metode za šifriranje i dešifriranje podataka gdje se atributima opisuju podaci i parametri, a metodama vraća šifrirani tj. dešifrirani sadržaj.
5. …

**2. Aplikacija sadrži minimalno tri dijaloga (forme) te demonstrira komunikaciju između minimalno dva dijaloga (npr. prijenos unesenih podataka iz jednog u drugi), 2 boda. U slučaju kreiranja i korištenja više formi moguće je dobiti 3 boda.**

Navedena funkcionalnost ovisi o tipu (svrsi) aplikacije, no najčešće svaki projekt ima minimalno tri dijaloga (forme). Primjerice, to mogu biti dijalozi za

1. prijavu korisnika u aplikaciju
2. glavni dijalog aplikacije
3. dijalog sa podacima o autoru aplikacije
4. razni dijalozi za unos i pregled podataka
5. …

Kao primjere prijenosa podataka iz jednog dijaloga u drugi, moguće je npr. unesene korisničke podatke sa „login“ dijaloga prenijeti i prikazati u glavnom dijalogu aplikacije gdje bi se uz korisničko ime prikazali i puni podaci o prijavljenom korisniku. Iz drugih dijaloga također je moguće prenijeti i neke druge podatke. Npr. razne postavke aplikacije koje su definirane u jednom „settings“ dijalogu (kao što su font i veličina teksta), koje se zatim mogu prenijeti tj. primijeniti na drugom (npr. glavnom) dijalogu aplikacije.

**3. Aplikacija podržava sučelje na barem dva jezika (npr. HRV i ENG), 3 boda. U slučaju da aplikacija ima 5 ili više dijaloga moguće je dobiti 4 boda.**

Ovisno o broju dijaloga (formi) u aplikaciji biti će potrebno više ili manje vremena za prijevod kompletne aplikacije. Prilikom prijevoda studenti mogu koristiti demonstrirani pristup tj. korištenje „Resource DLL Wizard“ čarobnjaka (C++ Builder), a mogu koristiti i druge pristupe (npr. dohvaćanje prijevoda koji je spremljen u bazi podataka ili nekoj drugoj datoteci).

**4. Za spremanje i učitavanje postavki aplikacije (minimalno 3) demonstrirati korištenje INI datoteke ili Windows registra (2 boda). U slučaju da aplikacija koristi oba pristupa (INI + Windows registar) moguće je dobiti 3 boda.**

Moguće je spremati razne postavke aplikacije:

1. Font i veličina teksta na dijalozima ili određenim komponentama
2. Korištene boje
3. Pozicije komponenti u dijalogu
4. Odabrane postavke u izbornicima
5. Personalizirane postavke za pojedinog korisnika aplikacija
6. …

U slučaju da se koriste oba pristupa (INI i Windows registar) svaki od njih treba demonstrirati nad zasebnim (različitim) nizom podataka. Npr. font, veličinu teksta i korištene boje u INI datoteci, a pozicije komponenti, odabrane postavke u izbornicima itd. u Windows registru.

**5. Za spremanje i obradu podataka u aplikaciji demonstrirati korištenje XML ili JSON tipa datoteka (3 boda). Potrebno je demonstrirati operacije čitanja, pisanja, uređivanja i brisanja podataka. U slučaju da aplikacija koristi oba pristupa (XML + JSON) moguće je dobiti 4 boda.**

Ovisno o problemu koji aplikacija (projekt) rješava podatke je moguće spremati, učitavati i obrađivati korištenjem raznih već postojećih tipova datoteka kao što su XML i JSON. Primjerice, to mogu biti nizovi podataka o stavkama unutar skladišta robe, akcijama (logovima) unutar aplikacije, rad sa šifrarnicima unutar aplikacije itd. U slučaju da se koriste oba pristupa (XML i JSON) svaki od njih treba demonstrirati nad zasebnim (različitim) nizom podataka. Npr. podatke o skladištu obrađivati korištenjem XML, a podatke o akcijama (logovima) korištenjem JSON tipa datoteka.

**6. Za spremanje i obradu podataka u aplikaciji razviti i koristiti datoteku prilagođenog (binarnog) formata. Potrebno je demonstrirati operacije čitanja i pisanja podataka.**

Uz korištenje nekih od već postojećih tipova datoteka (npr. XML i JSON) moguće je razviti i vlastiti format datoteke za čitanje i pisanje (spremanje) podataka. Za implementaciju ove funkcionalnosti obično je potrebno napisati dvije klase (klasa za zaglavlje i klasa za pojedini zapis), pa se te klase mogu samo jednom bodovati kao sastavni dio ove funkcionalnosti (neće se dodatno bodovati i kao sastavni dio funkcionalnost br. 1). Ukoliko je već implementirana funkcionalnost br. 5, tada ova funkcionalnost treba biti demonstrirana nad drukčijim nizom podataka.

**7. Aplikacija se spaja na bazu podataka po izboru te može izvršavati operacije čitanja, dodavanja, uređivanja i brisanja zapisa nad barem dvije tablice (3 boda). U slučaju da aplikacija koristi 3 ili više tablica nad kojima se demonstriraju opisane funkcionalnosti moguće je dobiti 4 boda.**

Funkcionalnost se može implementirati kao alternativa ili dodatak funkcionalnostima 5 i 6. Primjerice, moguće je imati implementirane sve tri funkcionalnosti (5, 6 i 7), ali svaka od njih mora biti demonstrirana nad različitim nizom podataka (npr. podatke o skladištu obrađivati korištenjem XML-a, podatke o akcijama (logovima) korištenjem JSON-a, a podatke o šifrarnicima korištenjem baze podataka).

**8. Aplikacija ima mogućnost sortiranja zapisa učitanih iz baze podataka, njihovog filtriranja, te koristi barem jedan izračunati (eng. *calculated*) ili *lookup* tip polja.**

Implementacija sortiranja i filtriranja sadržaja baze podataka ovisi o odabranom programskog jeziku i razvojnom okviru. Isto vrijedi i za izračunata i „lookup“ polja. Oba tipa polja ne postoje u bazi podataka (samo u memoriji aplikacije), a primjeri bi mogli biti sljedeći:

Ukoliko tablica *Student* sadrži sljedeće stupce:

* IDKolegij (int, strani ključ iz tablice *Kolegij*)
* Kolokvij1\_bodovi (int)
* Kolokvij2\_bodovi (int)
* Vjezbe\_bodovi (int)

Izračunato polje može biti *Kolokviji\_Ukupno* (Kolokvij1\_bodovi + Kolokvij2\_bodovi), *Bodovi\_Ukupno* (Kolokvij1\_bodovi + Kolokvij2\_bodovi + Vjezbe\_bodovi), te *Konacna\_Ocjena* (po nekoj formuli, ovisno o ukupnom broju bodova). Kao primjer „lookup“ tipa polja može biti polje *Kolegij* (string) koje će u tablici *Student* za vrijednost u stupcu *IDKolegij* dohvatiti naziv tog kolegija iz tablice *Kolegij*.

**9. Aplikacija može automatski generirati i prikazati barem jedan izvještaj. Izvještaj je moguće spremiti u PDF i RTF formatima.**

Ovisno o odabranom programskom jeziku i razvojnom okviru za generiranje izvještaja moguće je koristiti razne generatore. Najčešće je riječ o generatorima izvještaja koji podatke dohvaćaju iz baze podataka. Generirani izvještaji mogu prikazati razne setove podataka, a to su najčešće podaci dobiveni kao rezultat upita nad bazom podataka. Npr.

1. Izvještaj o stanju skladišta sortiran po stavkama (od onih koji ima najmanje prema više)
2. Izvještaj o prolaznosti studenata na ispitu sortiran po ocjeni
3. Izvještaj o najprodavanijim artiklima u (online) dućanu u nekom vremenskom periodu
4. …

**10. Aplikacija može paralelno izvršavati zadatke korištenjem dretvi. U aplikaciji je potrebno demonstrirati paralelno izvršavanje barem tri zadatka.**

Paralelno izvršavanje podrazumijeva da se u isto vrijeme izvršava više zadataka (više dretvi radi u isto vrijeme). Npr.,

1. Analiza podataka gdje više dretvi paralelno pretražuje isti popis zaposlenika s ciljem analize koliko tih zaposlenika su muškog i ženskog spola. Svaka od dretvi tada treba pretraživati svoj dio popisa zaposlenika umjesto da jedna (glavna) dretva pretražuje cijeli popis
2. Izračun ukupnog prosjeka svih ocjena studenata u popisu (nizu) gdje svaka od dretvi računa prosjek ocjena različitog studenta
3. Paralelno sortiranje više nizova podataka gdje se svaki od nizova sortira u zasebnoj dretvi
4. Paralelno spremanje više zapisa bazu podataka
5. …

**11. Aplikacija demonstrira sigurno ažuriranje komponenti korisničkog sučelja iz jedne ili više dretvi (npr. korištenjem sinkronizacije).**

Mogući primjeri:

1. Dretva dodaje izračunate podatke u Edit komponentu koja se nalazi na glavnom dijalogu.
2. Dretva nakon završetka svog rada mijenja stanje neke komponente na glavnom dijalogu (npr. uvećava poziciju ProgressBar komponente za 1)
3. Dretva u stvarnom vremenu na glavnom dijalogu prikazuje postotak izvršavanja svog zadatka (npr. preuzimanje datoteke sa Interneta)
4. …

**12. Aplikacija demonstrira korištenje barem jednog mehanizma međusobnog zaključavanja (npr. kritična sekcija ili mutex).**

Mogući primjeri:

1. Izračun prosjeka ocjena niza studenata gdje više dretvi za sumu svih ocjena koristi isti (zajednički) brojač
2. Više dretvi koristi isti (zajednički) brojač koji se uvećava za 1 svaki puta kada jedna od dretvi završi s radom (npr. za prikaz trenutnog broja završenih dretvi u ProgressBar komponenti)
3. Više dretvi sprema svoje logove ili rezultate izvršavanja u istu datoteku, gdje je u isto vrijeme dozvoljeno da u datoteku piše maksimalno jedna dretva (ostale čekaju u redu)
4. …

**13. Demonstrirati TCP klijent-poslužitelj komunikaciju u kojoj su klijent i poslužitelj implementirani u različitim aplikacijama. Potrebno je prikazati primjer u kojemu se kao zahtjev/odgovor prenose jednostavni tipovi podataka (3 boda). U slučaju i prijenosa tokova (eng. streams) moguće je dobiti 4 boda.**

Mogući primjeri funkcionalnosti za jednostavne tipove podataka:

1. Klijent šalje dva broja (double) i računsku operaciju (string), a kao odgovor dobiva rezultat te računske operacije
2. Klijent šalje JMBAG studenta i broj semestra, a kao odgovor od poslužitelja dobiva popis njegovih upisanih kolegija na tom semestru
3. Klijent šalje ISBN knjige, a kao odgovor dobiva naziv autora, izdavača i godinu izdanja knjige
4. ….

Mogući primjeri funkcionalnosti za prijenos tokova:

1. Prijenos datoteke sa klijenta na poslužitelj, gdje se datoteka arhivira na računalu poslužitelja
2. Klijent šalje dva broja A i B (A < B), a kao odgovor dobiva XML ili JSON datoteku koja sadrži sve prim brojeve u intervalu [A, B]
3. Klijent šalje XML datoteku s popisom studenata, a kao odgovor dobiva taj isti popis studenata u JSON datoteci (poslužitelj u svrsi konverzije podataka).
4. …

**14. Demonstrirati UDP klijent-poslužitelj komunikaciju u kojoj su klijent i poslužitelj implementirani u različitim aplikacijama. Potrebno je prikazati primjer u kojemu se kao zahtjev/odgovor prenose jednostavni tipovi podataka (3 boda). U slučaju i prijenosa tokova (eng. streams) moguće je dobiti 4 boda.**

Isto upute kao i u slučaju prethodne funkcionalnosti. Ovu funkcionalnost je moguće kombinirati sa prethodnom ukoliko je riječ o demonstraciji različitih primjera.

**15. Aplikacija sadrži HTTP klijent komponentu te demonstrira preuzimanje sadržaja s Interneta. Potrebno je prikazati tijek i postotak preuzimanja sadržaja te omogućiti ograničenje brzine na nekoliko proizvoljno unaprijed definiranih vrijednosti.**

Potrebno je odabrati datoteku koja se nalazi na HTTP poveznici (HTTPS neće raditi bez dodatne implementacije SSL/TLS protokola). Preuzeta datoteka treba imati svrhu tj. koristi od strane aplikacije. Kao demonstraciju moguće je primjerice prikazati:

1. Preuzimanje nove inačice vlastite aplikacije
2. Preuzimanje najnovije inačice baze podataka. Npr. najnoviji cjenik auto-dijelova (MS Access datoteka) za aplikaciju automehaničarske radionice
3. Preuzimanje dodatnog modula aplikacije (npr. DLL biblioteke sa novim funkcionalnostima)
4. Preuzimanje nekog drugog dodatnog sadržaja koji se može koristiti unutar aplikacije

**16. Aplikacija sadrži SOAP klijent komponentu pomoću koje se spaja na udaljeni (online) SOAP web servis po izboru. U aplikaciji je potrebno demonstrirati korištenje barem jedne funkcionalnosti tog web servisa.**

Unutar vlastite aplikacije potrebno je implementirati komunikaciju s nekim od vanjskih (već postojećih) SOAP web servisa na Internetu. Odabrani web servis bi trebao imati smisao tj. na neki način pomoći rješavanju problema kojim se bavi vaša aplikacija. Neki od već postojećih SOAP web servisa koji su trenutno aktivni su u nastavku.

<http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso>

<https://www.dataaccess.com/webservicesserver/NumberConversion.wso>

<https://www.dataaccess.com/webservicesserver/TextCasing.wso>

<http://webservices.daehosting.com/services/isbnservice.wso>

<http://soapclient.com/newsservice.html>

<http://www.learnwebservices.com/services/tempconverter?wsdl>

**17. Aplikacija sadrži SOAP klijent komponentu (2 boda) pomoću koje se spaja na vaš lokalni SOAP web servis implementiran u zasebnoj aplikaciji (2 boda). U aplikaciji je potrebno demonstrirati korištenje barem jedne funkcionalnosti tog web servisa.**

Vlastiti lokalni web servis bi trebao predstavljati klasu „u oblaku“ koja nudi metode koje se mogu koristiti u vašoj aplikaciji. Primjerice:

1. SOAP web servis koji nudi metode za analizu nekog niza podataka (npr. za zadani interval brojeva [A, B] vratiti koliko njih su djeljivi sa brojem X, koliko ih je prim brojeva itd.)
2. SOAP web servis sa metodama za čitanje baze podataka (npr. klijent šalje JMBAG studenta i broj semestra, a servis upitom prema svojoj bazi podataka vraća koliko je taj student do sada položio ispita u traženom semestru)
3. SOAP web servis koji nudi metode za konverziju podataka (XML -> JSON, JSON->XML itd.). Sadržaj jedne datoteke šalje se kao parametar (tekst, npr. XML), a kao odgovor se dobije također tekst tj. sadržaj datoteke u JSON formatu.
4. SOAP web servis koji vraća trenutni datum i vrijeme na poslužitelj računalu (npr. za potrebe sinkronizacije sa klijentima)
5. …

**18. Aplikacija sadrži REST klijent komponentu pomoću koje se spaja na udaljeni (online) REST web servis po izboru. U aplikaciji je potrebno demonstrirati korištenje barem jednog resursa tog web servisa.**

Slične upute kao i za funkcionalnost broj 16. Neki od već postojećih REST web servisa koji su trenutno aktivni su u nastavku.

<http://geodb-cities-api.wirefreethought.com/docs/api/get-city-date-time>

<https://api.ipify.org/?format=json>

<https://ipinfo.io/161.185.160.93/geo>

<https://official-joke-api.appspot.com/random_joke>

<https://apipheny.io/free-api/>

**19. Aplikacija sadrži REST klijent komponentu pomoću koje se spaja na vaš REST web servis implementiran u zasebnoj aplikaciji. U aplikaciji je potrebno demonstrirati korištenje barem jednog resursa tog servisa s barem dvije metode (npr. GET i POST).**

Mogući primjeri:

1. REST web servis koji korištenjem resursa *student* i metode GET za predani JMBAG studenta i broj semestra upitom nad bazom podataka vraća broj njegovih položenih ispita u tom semestru. U slučaju korištenja metode POST treba se u obliku XML datoteke vratiti poruka o grešci („metoda nije podržana!“).
2. REST web servis koji korištenjem resursa *isbn* i metode GET za predani ISBN iz baze podataka dohvaća i vraća naziv knjige, izdavača i datum izdavanja. U slučaju poziva metode POST ti isti podaci se kao novi zapis spremaju u lokalnu bazu podataka web servisa.
3. REST web servis koji korištenjem resursa *datetime* i metode GET vraća trenutni datum i vrijeme na poslužitelj računalu (npr. za potrebe sinkronizacije sa klijentima). Parametrom *format* moguće je definirati da li se rezultat vraća u XML ili JSON formatu, ili kao običan tekst.
4. REST web servis koji korištenjem resursa *student* i metode GET za predani JMBAG vraća sve podatke o tom studentu iz baze podataka, a metodom PUT sprema podatke o novom studentu u bazu podataka.
5. …

Ovu funkcionalnost je moguće kombinirati sa funkcionalnosti broj 17 ukoliko je riječ o demonstraciji različitih primjera.

**20. Aplikacija ima mogućnosti šifriranja i dešifriranja sadržaja korištenjem simetričnog kriptografskog algoritma (npr. AES).**

Šifriranje i dešifriranje je moguće upotrijebiti u raznim situacijama. Primjerice, šifriranje i dešifriranje korisničkog imena spremljenog u bazi podataka, šifriranje raznih postavki aplikacije koje su spremljene u datotekama, podataka koji se prenose u klijent-poslužitelj komunikaciji (klijent šalje šifrirani sadržaj koji se dešifrira na poslužitelju), šifriranje datoteka itd.

**21. Aplikacija ima mogućnosti šifriranja i dešifriranja sadržaja korištenjem asimetričnog kriptografskog algoritma (npr. RSA).**

Isto upute kao i u slučaju prethodne funkcionalnosti. Ovu funkcionalnost je moguće kombinirati sa prethodnom ukoliko je riječ o demonstraciji različitih primjera.

**22. Aplikacija koristi barem jednu funkciju sažimanja (npr. SHA-256).**

Mogući primjeri:

1. Provjera lozinke korisnika
2. Provjera integriteta poruke ili datoteke. Npr. klijent šalje poruku (tekst) ili datoteku mrežom, a poslužitelj provjerava njenu očekivanu hash vrijednost te na taj način detektira ukoliko je poruka ili datoteka mijenjana tokom prijenosa mrežom.

**23. Aplikacija ima mogućnost digitalnog potpisivanja sadržaja. U aplikaciji je potrebno demonstrirati verifikaciju autora i primljenog sadržaja pomoću digitalnog potpisa.**

Mogući primjeri:

1. Izvoz podataka iz baze podataka u digitalno potpisanu XML/JSON datoteku, te uvoz podataka u bazu podataka iz digitalno potpisane XML/JSON datoteke (uz prethodno uspješnu verifikaciju digitalnog potpisa).
2. Chat komunikacija između klijenta i poslužitelja gdje klijent šalje poruku (tekst) ili datoteku mrežom te njen digitalni potpis. Poslužitelj odgovara na zahtjev klijenta porukom „OK“ ukoliko utvrdi da je digitalni potpis sadržaja valjan, ili porukom „INVALID“ ukoliko nije.
3. Preuzimanje datoteke sa Interneta, gdje se uz datoteku preuzima i njen digitalni potpis. Provjerom digitalnog potpisa aplikacija obavještava korisnika da li je riječ o datoteci s izvornim sadržajem.

**24. Aplikacija koristi programski kod implementiran u statičkoj biblioteci (minimalno dvije funkcije), 2 boda. U slučaju da biblioteka sadrži barem jednu klasu moguće je dobiti 3 boda.**

Navedenu funkcionalnost moguće je kombinirati sa funkcionalnosti broj 1.

**25. Aplikacija koristi programski kod implementiran u dinamičkoj biblioteci (minimalno dvije funkcije), 2 boda. Način povezivanja aplikacije i dinamičke biblioteke je proizvoljan. U slučaju da biblioteka sadrži barem jednu klasu moguće je dobiti 3 boda.**

Isto upute kao i u slučaju prethodne funkcionalnosti. Ovu funkcionalnost je moguće kombinirati sa prethodnom ukoliko je riječ o demonstraciji različitih primjera.

**26. Aplikacija koristi resurse smještene u dinamičkoj biblioteci.**

Moguće je izraditi vlastiti DLL sa resursima (npr. slikama), ili koristiti neke od već gotovih resursnih DLL biblioteka. Jedna od gotovih je i biblioteka CARDS.DLL (dostupna na LMS-2020 stranicama kolegija) koja u obliku BITMAP resursa sadrži karte za igru. Drugi mogući primjeru je spremanje slika (npr. logo tvrtke) ili spremanje drugih dijaloga (formi) kao resursa u DLL biblioteku, a zatim korištenje istih u svojoj aplikaciji (potrebno demonstrirati).

**27. Aplikacija koristi programski kod implementiran u zasebno razvijenoj komponenti. Komponenta treba sadržavati bar dva svojstva i barem jednu metodu.**

Razvijena komponenta bi se trebala moći koristiti u jednom od navedenih slučajeva:

1. U vrijeme dizajna aplikacije kao vizualna komponenta (npr. nova vlastita gumb komponenta)
2. U vrijeme dizajna aplikacije kao ne-vizualna komponenta (instanca klase)
3. Kao ActiveX

Mogući primjeri:

1. Komponenta „TempKonverzija“ koja ima svojstva *celsius* i *kelvin* te metode *celsiusToKelvin* i *KelvinToCelsius* koje vraćaju predviđene vrijednosti*.*
2. Komponenta „TezinaKonverzija“ koja ima svojstva *kg* i *lbs* te metode *kgToLbs* i *lbsToKg* koje vraćaju predviđene vrijednosti*.*
3. Komponenta „Log“ koja ima svojstva *datum,* *vrijeme* i *logDatoteka* te metodu *SpremiLog(string tekst)* koja zadani tekst (log) sa podacima o datumu i vremenu sprema u ciljanu datoteku.

Opisane funkcionalnosti nisu konačne i svi studenti mogu smisliti te predložiti i vlastite funkcionalnosti. Prije implementacije takvih funkcionalnosti potrebno je javiti se nastavniku glede odobrenja i bodovanja iste.