Rendszerterv

1. Funkcionális terv

1.1. Feladat leírása:

A feladat egy GPS-képes eszközökön futó alkalmazás, illetve ennek szerver oldali párjának létrehozása. A program a szerveren tárolt adatbázis alapján küldene információt a felhasználóknak. Egy felhasználó egy GPS-t ismerő eszköz lehet, de ezen belül nincs megkötés, tehát elvileg lehet ez mobiltelefon, PDA, laptop, stb. A felhasználó elküldi GPS-en keresztül megszerzett pozícióját a szervernek, mely egy térképrészletet küld vissza a felhasználónak, természetesen a kapott adatoktól függően (például egy teljes Magyarország térképet feltételezve a felhasználónak egyszerre maximum egy kisebb városnyi térképre lehet szüksége, ennél nagyobb, áttekintőbb térkép csak ritkán szükséges). A rendszer alapvetően többfelhasználós, vagyis nem csak egy felhasználó kommunikál a szerverrel, hanem egyszerre akár több is. Így lehetséges olyan megoldás is, hogy a felhasználók csoportokba sorolódnak, és egy csoporton belüli felhasználók látják egymás pozícióját a kapott térképen. Lehetséges továbbá olyan statikus „felhasználók" jelenléte is, akik nem igazi felhasználói a rendszernek, csak egyetlenegyszer kerül be a pozíciójuk a rendszerbe, majd ezt később kívánságra bármely felhasználó láthatja (a térképi esetben ez például jelenthet éttermeket, mozikat, bármilyen fix helyzetű, a felhasználó számára fontos helyet).

1.2. Rendszer célja, motivációja:

A feladat az önálló labor VIII. szemeszterének alapját képezi. Ennek megfelelően nem azok a követelmények vonatkoznak rá, mint egy a való életnek készülő projekt esetében. A cél itt az, hogy egy diák esetében meg lehessen ítélni, hogy egy (két) félév alatt mit képes önállóan alkotni. Már a tárgy neve is mutatja, hogy ez egy önálló munka lesz, amibe természetesen a konzulensek munkája is beletartozik, de mégis igazából egy ember teljesítményét kell ez alapján értékelni. Emiatt maga a projekt sem lehet olyan komplikált és mindenre kiterjedő, mint egy való életnek szóló munka. Ez elsősorban abban mutatkozik meg, hogy sok dolgot el lehet és el is kell hanyagolni, mert mindenre nem juthat idő. Emiatt fontos, hogy ki tudjuk emelni azokat a sarokpontjait egy projektnek, amit egy ember is el tud végezni ennyi idő alatt, de mégis egy egész áll belőle össze, ráadásul még értékelhető is kell, hogy legyen.
A projektet nem az üzleti szférának tervezzük és kivitelezzük. Nem kell foglalkozni egyelőre gazdaságossági és marketing elemekkel, a külcsín is mellékes még ezen a szinten. De emellett olyan területekkel is csak érintőlegesen foglalkozunk, mint például a rendszer integrálhatósága, az emberi felhasználásból fakadó problémák, vagy az üzemeltetés problémái. A hangsúly abba az irányba tolódik el, hogy láthatóvá váljon a hallgató felkészültsége azokból a technológiákból és paradigmákból, melyek az alapjait képezik egy ilyen rendszernek. Mindazonáltal érdemes figyelembe venni azt a tényezőt, hogy az elkészült projektet kis ráfordítással át lehessen vinni a való életbe is, hiszen mégiscsak ezzel fogunk foglalkozni a későbbiek folyamán, így nem árt ilyen téren sem a tapasztalatszerzés. Viszont fontos megtalálni a megfelelő arányt, hogy egy kicsit konkrétabb legyek, nem szabad eltölteni két hetet az amúgy is szűk időből a grafikai látványterv kidolgozására, téve mindezt például a tesztelési terv rovására.

1.3. Szereplők és igényeik:

A rendszert alapvetően két szereplőre lehet lebontani, viszont közülük csak egyikük igazi felhasználója a programnak, mivel a kliens szerver architektúrát feltételezve csak a kliens valódi felhasználó. A szerver oldal ugyanis el van rejtve a felhasználók elől abban az értelemben, hogy csak a fejlesztő fér hozzá, a valódi felhasználók nem. Ők egy kliens oldali alkalmazást futtatnak a megfelelő eszközükön. Mégis meg kell azonban különböztetni egy másik típusú felhasználót. Ők azok, akiket a feladatleírásban statikus „felhasználónak" neveztem, vagyis akik csak információt (pozícióadatokat) szolgáltatnak bizonyos, a valódi felhasználók érdeklődésére számot tartó helyekről. Nyilvánvalóan az ő igényeiknek is meg kell tudni felelni, mert egy esetleges valós életbeli hasonló projektben ők jelenthetik a bevétel egy jelentős részét, hiszen ez tulajdonképpen egy hirdetési felület lehet számukra.

1.3.1. Valódi felhasználók:

A legfontosabb, amit egy valódi felhasználóról megjegyezhetünk, hogy mobil eszközt fog használni, ennek minden előnyével és hátrányával együtt. Legfontosabb paramétere ezeknek, hogy viszonylag kis sávszélességen tudnak csak kapcsolatot teremteni más eszközzel. Az elsődleges korlátunk tehát a sávszélesség lesz. Mivel emberek fogják használni a kliensünket, ezért fontos, hogy a kezelése emberléptékkel számolva gyors legyen. Ez azt jelenti, hogy nem szívesen várunk hosszú, 5-10 másodperceket egy utasítás végrehajtására. Viszont arra szinte mindenki hajlandó, hogy egy alkalmazás futása során egyszer, de csak egyszer a futás elején várjon egy hosszabb időtartamot. Ezt az időtartamot kell kihasználnunk minden nagyobb méretű adat átvitelére, ezután már törekedni kell, hogy minimalizáljuk az adatok áramlását a kliens és a szerver között, előnyben részesítve a kliensben végzett műveleteket.
Azt is meg kell jegyezni, hogy ezek az eszközök általában kis számítási kapacitással rendelkeznek, ez alól azért a laptopok kivételek, viszont a GPS-szel ellátott eszközök között kisebb a részarányuk, és fejleszteni mindig a legrosszabb esetre kell. Természetesen ez csak viszonylagos, de a szerverhez képest mindenképp jelentősnek tekinthető a különbség. Emiatt meg kell fontolni például, hogy egy műveletet elvégeztessünk a klienssel, vagy küldjük el kérésként a szervernek, majd az csak a választ küldje vissza.
A fenti két megfontolás miatt egyelőre a tervezés ezen fázisában még nem zárható ki egyik módszer sem a kliens és a szerver közötti feladatmegosztással kapcsolatban. Tehát a tervezés és főleg a komponensek esetén figyelembe kell venni azt az esetet, amikor a kliens csak kérést küld a szervernek, és onnan válasz érkezik (gyakorlatilag távoli eljáráshívás vagy webservices), és azt az esetet is, amikor a kliensre nagyobb mennyiségű adatot egyszerre letöltünk, és később ezen végzünk különböző műveleteket. Az első megoldás nyilván jobb, mivel a nagyobb teljesítményű szerverre bízza az adattárolást és a feldolgozást is, viszont a sávszélesség szűk keresztmetszete miatt a második eset is szóba kerülhet.
A harmadik nem elhanyagolható momentum, hogy az ilyen eszközök megjelenítőképessége szintén korlátozott (természetesen a laptopok itt is kivételek). Ez számunkra előny, mivel nem kell komoly grafikai teljesítmény a megjelenítéshez illetve kisebb adatmennyiség átvitele is elegendő a kijelző maximális kihasználásakor is. Abból a szempontból viszont hátrány, hogy a grafikus felület tervezésekor vissza kell fogni magunkat, egy kis felületre nem zsúfolhatunk be túl sok információt.
Összességében elmondható, hogy egy kompaktul megtervezett felhasználói felületet kell a kliensnek készíteni, melyen könnyen és értelmesen lehet navigálni és a különböző funkciókat kihasználni.

1.3.2. „Hirdetők”:

Ezekkel a felhasználókkal kevesebb a „probléma”, hiszen az általuk támasztott követelmények lényegesen egyszerűbbek, mint az előző esetben. Felhasználói felületként a legegyszerűbben egy webes felület képzelhető el, melyen keresztül be tudnak jelentkezni a rendszerbe, és be tudják vinni az adataikat. Jelen esetben, mivel nem komplett rendszert tervezek, el lehet tekinteni olyan megoldások megvalósításától, mint a számlázási adatok gyűjtése és egyéb, a való élettel kapcsolatos adminisztratív problémák kezelése.

1.3.3. Szerver oldal:

A szerver oldalon nem beszélhetünk igazi felhasználóról. A fejlesztő elkészíti az alkalmazást, majd innentől kezdve csak akkor kell hozzányúlni, ha szükséges. Természetesen egy felhasználói felület azért nem haszontalan, ezen keresztül a karbantartást könnyebben meg lehet oldani. Ebben azonban a követelmények sokkal inkább a funkcionalitást követelik meg, mint sem a megjelenést.

1.4. Komponensek, erőforrásaik, interfészeik:

1.4.1. Kliens oldali komponensek listája

1.4.1.a Kapcsolatfelépítés: ez a komponens valószínűleg egy előre megírt és tesztelt komponens lesz, mivel ez egy igen gyakran használt része a különböző programoknak. Ami egyéni, azok a kapcsolatfelvételkor elküldött adatok lennének. Nyilvánvalóan első lépésben el kell küldeni a megszerzett GPS koordinátákat, de ezen kívül az eszközön való megfelelő futtatás érdekében olyan adatokat is el kellene küldeni, mint például az eszköz felbontóképessége, maximális memóriája, stb.

1.4.1.b. Megjelenítés: mivel Java alapon létezik a MapViewer alkalmazás, ezért valószínűleg ennek egy tárolt változatát fogom a megjelenítésre használni. A komponens bemenete egy adatbázis lekérdezés eredményéből származó térképformátumú fájl lesz, amely majd a szervertől érkezik.

1.4.1.c. Kontroller: ez a komponens kezelné a különböző felhasználói interakciókat, és ezeknek megfelelően cselekedne. Ilyen interakciók lennének például térképen való mozgás négy irányban, megjelenítési rétegek változtatása (utak, domborzat, stb.), többfelhasználós rendszerben a többi felhasználó megtekintése, legrövidebb út keresés hozzájuk, stb.

1.4.1.d. GPS: valószínűleg ezt is előre megírt változatban fogom átvenni, mivel ennek a technológiai háttere nem témája ennek a feladatnak, így ebben nem is szeretnék egyelőre elmélyedni. Mindazonáltal arra azért figyelni kell, hogy az így lekérdezett adatok kompatibilisek legyenek az Spatial koordinátarendszereinek valamelyikével, hogy az együttműködés megoldható legyen.

1.4.2. Szerver oldali komponensek listája

1.4.2.a. Kapcsolatfelépítés: ez a komponens valószínűleg egy előre megírt és tesztelt komponens lesz, mivel ez egy igen gyakran használt része a különböző programoknak. Az egyediség itt is jelentkezik abban a formában, hogy a kapcsolatfelépítéskor átvett adatok sajátosak. Itt kell tehát gondoskodni a koordináták megfelelő kezeléséről, és ha kell konverziójáról is. Továbbá fontosak az eszköz paraméterei is, amik a későbbi működést jelentősen befolyásolják.

1.4.2.b. Hirdetői oldal: egy egyszerű JavaServlet alkalmazás lenne, ami arra alkalmas, hogy a hirdetők tudjanak regisztrálni, megadni a saját pozíciójukat illetve a hirdetett hely egyéb adatait (pl. cím, elérhetőség, weblap, stb.). Nem terjedne azonban ki különböző, egyébként egy valós alkalmazásban fontos elemekre, mint például a fizetési paraméterek és a számlázás lehetőségei.

1.4.2.c. Lekérdező: gyakorlatilag a szerver oldal lelke lenne. Innen érnénk el elsősorban az adatbázist, ez jelentheti akár egy hirdetés tartalmának lekérdezését, egy felhasználó pozíciójánaklekérdezését vagy akár térképi lekérdezést is a Spatial adatbázisban. Tulajdonképpen tárolt, paraméterezhető SQL lekérdezések gyűjteménye.

1.4.2.d. Módosító: hasonló lenne az előző komponenshez, annyi változtatással, hogy itt nem lekérdező, hanem módosító SQL utasítások lennének, továbbra is paraméterezhető módon. Tipikus felhasználása például egy új hirdető felvétele a rendszerbe vagy egy új felhasználó belépése.

1.4.2.e. Webservices: igazából ez jelentené a kapcsolatot a kliens és a szerver között a kapcsolatépítés után. Ez azt jelenti, hogy a kliens egy szolgáltatást kér a szervertől, amit az egy tárolt eljárás formájában hajt végre, majd annak eredményét visszaküldi a kliensnek. Természetesen itt inkább csak a kisebb adatmennyiség átvitelét jelentő kérésekről lehet elsősorban szó, mivel használat közben a nagyobb adatcsomagok átküldése jelentősen lassítja a rendszert a sávszélesség szűk keresztmetszete miatt.

1.5. Környezet:

A programozási és implementációs környezet nagyrészt a feladat kiírásából fakadóan adott. Így például a program mögött elhelyezkedő adatbázisokat az Oracle 11g rendszerben fogom tárolni, erősen kihasználva a Spatial kiegészítés adta lehetőségeket. Az alkalmazás forráskódja szintén elég egyértelműen Java lesz, mivel a legtöbb mobil eszköz támogatja ezt a platformot, így nem lesz nehéz a fejlesztést áttenni rájuk. További előny, hogy az Oracle beépítetten kínál Java fordítót (JDeveloper), mely így még kényelmesebbé teszi a munkát. Ezenkívül az Oracle Spatial szerves kiegészítője a MapViewer alkalmazás, mely a térképek megjelenítését teszi lehetővé. Ezt is Java-ban írták, ráadásul a JDeveloper is támogatja a használatát. A Java alkalmazása még azzal az előnnyel is jár, hogy a kliens-szerver architektúra megvalósítása sem jelent nagy nehézséget, mivel ez funkció a Java-ban egyszerűen megvalósítható, így a kommunikációs háttér megvalósítása sem fog túl nagy feladatot jelenteni. Java-ban továbbá van lehetőség webes felületek implementálására (Java Servlet) is, így egy felületen meg tudom oldani a különböző felmerülő problémákat.

Mivel a Java alapvetően platformfüggetlen programozási nyelv, így a szoftvereket futtató operációs rendszer típusa csaknem lényegtelen. Azért nem szabad ezt a paramétert sem elhanyagolni, de nagy vonalakban elmondható, hogy a GPS tudással rendelkező mobil eszközök nagy része Windows vagy Symbian alapú operációs rendszert használ, továbbá a fejlesztés alatt én is a Windows-t fogom előnyben részesíteni. A szerver oldali alkalmazás futása elsődlegesen ugyanazon a gépen fog történni, ahol a fejlesztés is, emiatt itt problémák talán csak kisebb mértékben fognak jelentkezni. A későbbiekben lehet szó nyilvános webszerveren való üzemelésről is, ez azonban még a jövő tárgya egyelőre.
Mivel nem rendelkezem egyelőre olyan eszközzel, ami GPS tudással rendelkezne, ezért valószínűleg valamilyen, az Interneten fellelhető emulátort fogok majd használni a teszteléshez.