Факултет техничких наука у Новом Саду

Елементи развоја софтвера

Техничка документација за пројекат: ‘Asset Management’ - Апликација за управљање имовном

Професор:
Варга Ервин

Асистент-ментор:
Бабић Зорана

Пројекат радили студенти:
Орестијевић Јована
Делић Стефан
Буха Тамара
Стаменковић Сара

Јануар 2023. године

Садржај техничке документације

1. Захтев за пројектовање

1.1. Опис проблема

Потребно је направити дизајн система, архитектуру система, имплементирати и истестирати решење који симулира рад и комуникацију Ассет Манагемент система. АМС води рачуна о свим уређајима у систему као што су на пример: прекидачи, трансформатори, осигурачи, вентили, генератори… и осигурава њихов стабилан рад пратећи број извршених операција и број радних сати.

1.2. Циљеви пројекта и захтеви

Основни циљ овог система је правилно одржавање опреме. Пројекат се састоји из дзајна система, архитектуре система, имплементације и тестирања решења који симулира рад и комуникацију управљача имовине (енг .‘Asset Management’). Управљач имовином представља систематично коришћење и праћење чинилаца имовине неког система, организације, декларације, скупине, простора или објекта. Обједињује ентитете од важности за уређаје, који исте прате читав њихов радни век (или док се не замене неком оптимизованијом варијантом - новим моделом/генерацијом уређаја). Постоји више типова АСМ-ова, овај конкретан пројекат симулира рад простих и сложених електромагнетичких и електротехничких уређаја, као и уређаја за регулацију протока течности и/или гаса, које често видимо и користимо у свакодневном животу. Неки од њих су: прекидачи, утикачи, трансформатори, генератори, вентили, осигурачи...
Циљ пројекта је веродостојно, једноставно, једнознчно и модерно симулирање АСМ-а за једноставне уредјаје у домаћинству.

Захтеви:

Систем садржи 3 компоненте:

1. Локални уређај
2. Локални контролер
3. Asset Management (AMC)

Локални уређај је једно мерно место у електроенергетском систему. Локални уређај може да мења стање на два начина:

- Дигитално (ОН/ОФФ, ОПЕН/ЦЛОСЕ...) - прекидачи, осигурачи, вентили итд.
- Аналогно (сетпоинт) - генератори, батерије итд.

Локални уређај сваку промену шаље локалном контролеру или директно АМС, у зависности од подешавања локалног уређаја:

- ‘Local device code’ - јединствено име уређаја имплементирано као хасх цоде
- ‘Timestamp’ (‘UNIX’ тиместамп формат)

- ‘Actual value’ (тренутна вредност, опен, цлосе, он, офф, аналог меасуремент)

Апликација Локалног уређаја је засебна конзолна апликација. Локални уређај се пали ручно из апликације и може бити угашен у сваком моменту, како плански из апликације тако и непланski гашењем саме апликације (тиме се симулира отказ опреме). Додавање новог Локалног уређаја се ради по принципу плуг-анд-плау, што значи да када се нови Локални уређај упали (упали се нова инстанца конзолне апликације), почиње слање својих података и мора бити прихваћено од стране Локалног контролера или АМС осим у случају ако то име већ постоји у систему. Слање података је периодично а број секунди трајања циклуса дефинише се у ХМЛ конгфигурационом фајлу.

Локални контролер чува сва промене која долазе од стране свих локалних уређаја пријављених на контролер и на сваких 5 минута (време је конфигурабилно у ХМЛ фајлу) их прослеђује АМС-у. У случају успешног слања Локални контролер брише своју бафер базу (ХМЛ), а у случају неуспешног чува бафер до успешног слања. Ако се апликација насилно угаси пре слања бафера, приликом иницијализације учитаће се вредности из фајла.
Локални контролер може бити упаљен у сваком моменту, али може бити и угашен исто као и локални уређај.

Апликација ЛК-а је засебна конзолна апликација која своју базу чува у ХМЛ фајлу. Додавање новог ЛК-а се ради по истом принципу као и додавање уређаја. У систему може постајати више ЛК апликација.

‘AMS’- чува све промене у систему у својој бази која је јединствена за цео систем и служи за прављење извештаја:

- Детаљи промена за изабрани период за изабрани локални уређај (све промене + сумарно);
- Број радних сати за изабрани уређај за изабрани временски период (од - до календарски по сатима);
- Излиставање свих уређаја чији је број радних сати преко конфигурисане вредности (алармирати и обојити у црвену боју оне уређаје за које је број радних сати већи од границе дефинисане у опцијама апликације);
- Листање свих постојећих уређаја у систему.

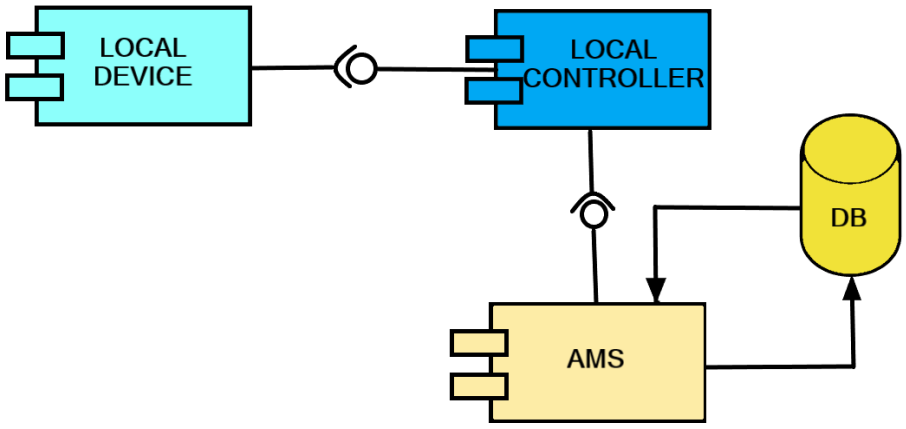
АМС апликација је засебна апликација која има свој кориснички интерфејс (може бити и терминал) и своје податке чува у ‘SQL’ бази.
Када се направи нови локални уређај, у конфигурацији се бира ком локалном контролеру или АМС припада, па стога мора да се излиста списак свих ЛК и конкретног система приликом креирања уређаја.

1.3. Очекивани напредак

Очекивани напредак израде пројекта и подела послова представљени су кроз 4 спринта који прате техничко решење. Техничко решење ће бити објашњено у наредним поглављима.
Тим који ради на пројекту састоји се од четири члана (студената који су претходно наведени као аутори и ове документације).
Како је овај проблем уско везан са реализацијом самог курса ‘Елементи развоја софтвера’, пројекат прати последњих пар недеља курса (семестра) и почетну испину недељу - у којој је одбрана истог.

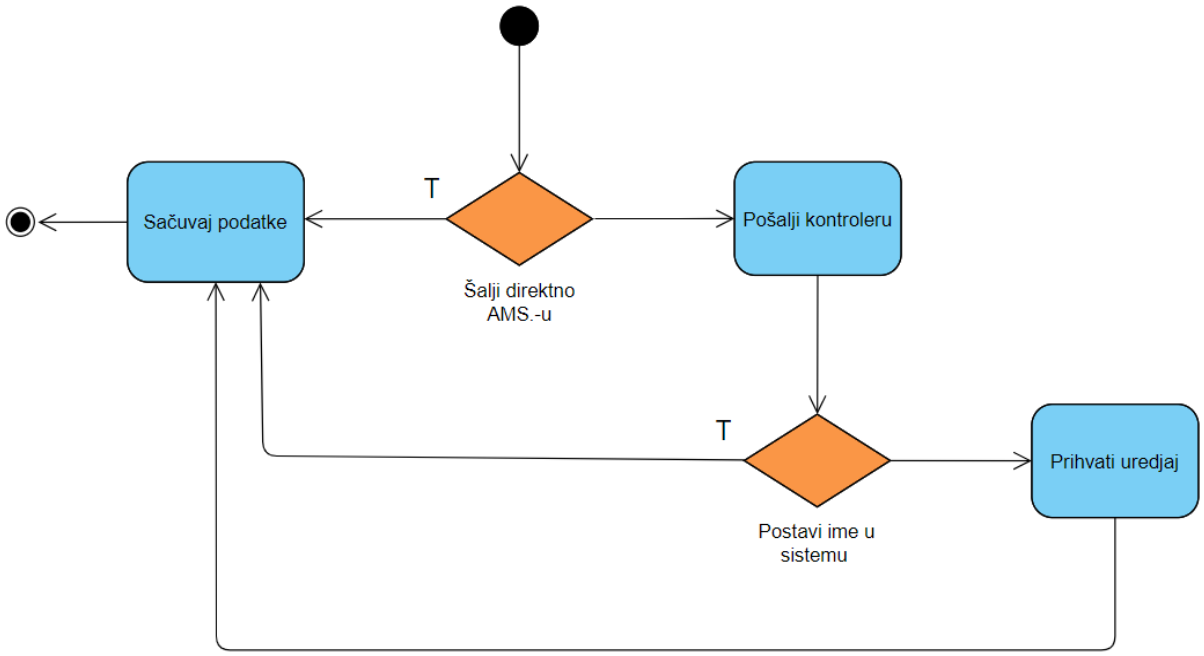
2. Идејно решење

2.1. Дијаграм компоненти



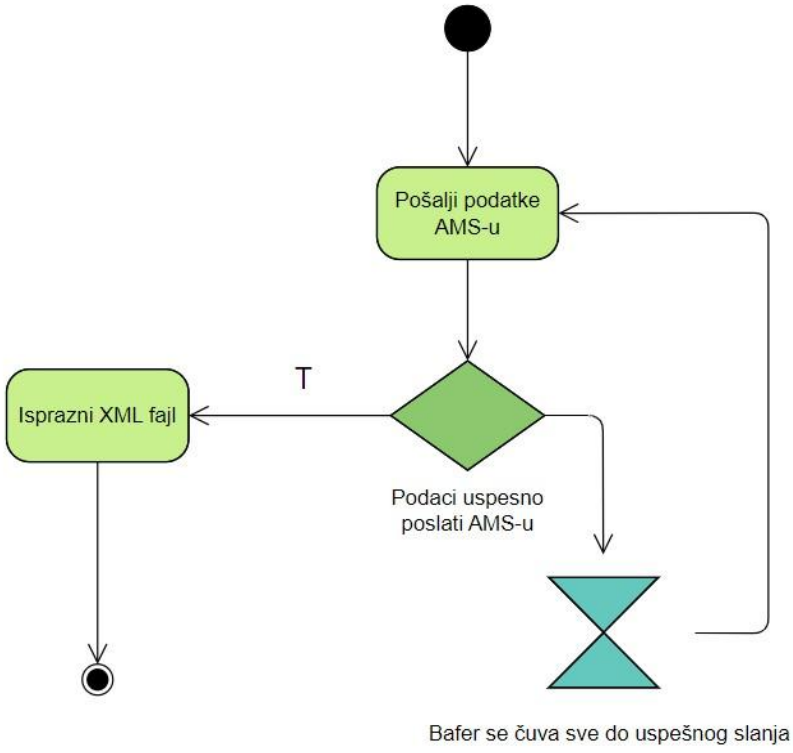
Објашњење дијаграма…

2.2. Дијаграм активности слање подтака о уређају



Објашњење дијаграма…

2.3. Дијаграм активности слање подтака од контролера ка AMC-у



Објашњење дијаграма…

3. Техничко решење

3.1. Списак активности

- **‘LocalDeviceProject’** aplikacija
  - Укључује интерфејсе: **ILocalDevice** и **IMyNetworkStream**
  - Испис активности (додавање, да ли је послато, тренутно станје…) у конзоли
  - Креирање конструктора класе LocalDevice
  - Оверрајдовање **ToString()** методе
  - Креирање хеш вредности
  - **NetworkStream Stream** објекат који обезбеђује методе за слање и пријем података преко Стреам сокета у режиму блокирања. Уз ппомоћ њега мозе се вршити и синхрона и асинхрона комуникација.
  - **TCP (client-server)** комуникација са AMS-om и са локалним контролером

        - Покретање комуникације - **Startup()**
        - Слање података - **SendData()**

- **‘LocalControllerProject’** апликација
    - Укључује интерфејс **ILocalController**
    - Након успешне конекције рпихвата и креира сопствен објекат клијента **ReceiveData()**
        - Бинарно форматира и меморишки стримује податке уз помоћ функција из **BinaryFormatter**
    - Прављење клијента са којим цемо даље радити - **Startup()**
        - **MyTcpClient**
    - Покретање новог направљеног сервера - **StartServer()**
        - Уклјучује два ‘**MyNetworkStream**’ - један за локални контролер један за AMC
        - Ослушкује долазак потенцијалних клијената - **MyTcpListener**
        - Започиње рад - **StartServer()** → све док стартовани сервер ради прихватају се подаци и уписују у XmlWriter
    - У комуникацији са AMCом шаље податке SendToAMS()
        - Прави листу уређаја од ппрочитаних података из XmlReader (уз помоћ **ReadData()** функције)
        - Бинарно форматира и меморишки стримује податке уз помоћ функција из BinaryFormatter
        - Уписује податке у ‘**MyNetworkStream**’ објекту направљеном у **StartServer()** за AMC
    - Функционалности и имплементацију за **MyTcpListener** и **MyTcpClient** класе коришћене у претходим класама
    - Функционалности за **XmlReader** и **XmlReader** класе коришћене у претходим класама

- ‘’AssetManagement’’
    - AMS
        - Покретање сервера - AMS апликација представља крајњи сервер чији клијент могу бити уређај или локални контролер (или више њих) - **StartServer()**
        - **KomunicirajSaUredjajima()** → привата листу коју ја преходно направљена у **IspisiSveUredjaje()** → користи се као интеративни пролазак до краја листе која је послата као аргумент, који након тога прави нову листу (која постојји само у тој функцији) прођлих уређаја из прослеђене листе. Тако добијамо тренутни објекат који принтујемо у конзоли проласком кроз тренутну листу и исписом елемената на конзоли итеративним путем.
        - **DaLiPostoji()**
            - проверава да ли проследјен уредјај постоји у листи

            - Користи се у **IspisiSveUredjaje()** како би иницирао прескаканје у проласку кроз листу → зато што исти већ постоји
        - Функција која прихата податке од клијента, даје обавештење о томе и исписује их → **ReceiveData()**
            - Креира клијента и проверава успешност конекције са тренутним клијентом
            - Креира празну листу која ће се у наставку користити за лакши рад и чување базе
            - Бинарно форматира и меморишки стримује податке уз помоћ функција из BinaryFormatter и MemoryStream
    - DataBase
        - Слање команди
        - Унос података у базу
        - Виртуалне методе sa креиранје и чување
            - У овом пројекту SQL бази приступа и за њу се директно везује само AMC, а ‘’AssetManagement’’ пројекат обједињује све остале пројекте, ове методе су виртуалне да се не би бунио пројекат приликом рада програма → ово није најбоље решење али бes овога тестови би можда правили проблем.
            - Виртуелне методе дозвољавају подкласама типа да замене метод. Користе се за имплементацију полиморфизма времена извршавања или касног везивања. Када је метода декларисана као виртуелна у основној класи, а иста дефиниција постоји у изведеној класи, нема потребе за заменом, али другачија дефиниција ће функционисати само ако је метода замењена у изведеној класи.

- **‘’AssetManagementTest’’**
    - Тестирање претходне три апликације → биће обрађено у поглављу
    - Уклјучује:
        - **AMSTests**
        - **LocalControllerTest**
        - **LocalDeviceTest**