Универзитет у Београду,

Електротехнички факултет

Пројекат из конкурентног и дистрибуираног програмирања:

Сихнронизација докумената на већем броју рачунара

Студент:

Ђорђе В. Димитријевић 2018/0489

Ментор:

Сања Делчев

Београд, Електротехнички факултет, 24.2.2021.

Садржај

Текст пројекта:

Пројектовати дистрибуирани рачунарски систем који треба да омогући синхронизацију докумената на већем броју рачунара. Програм треба да ради у систему који се састоји од више рачунара повезаних у LAN (Local Area Network) или WAN (Wide Area Network). У систему постоји три типа програма:

- Централни сервер који служи за чување оригиналних верзија докумената и информација о распрострањености копија докумената.
- Подсервери на којима се чувају копије докумената, чији се оригинали налазе на централном серверу.
- Клијентски програм преко кога клијент може да приступа документима на подсерверима.

Када клијентски рачунар пошаље захтев за неким документом неком од подсервера, тај подсервер проверава да ли се тражени документ налази код њега. Уколико се документ налази на подсерверу, тада подсервер кориснику враћа тражени документ. Уколико се документ не налази на подсерверу, онда подсервер од централног сервера тражи да му пошаље копију документа. Када на подсервер пристигне копија документа, она се памти на подсерверу, а такође се прослеђује и кориснику који је тражио документ.

Централи сервер периодично мења садржаје оригиналних докумената. Потребно је обезбедити одржање лабаве конзистентности на следећи начин. Када централни сервер утврди да неки документ треба променити он интерно провери на којим се све подсерверима налази тражени документ. Када је сервер утврдио листу подсервера, онда шаље свим подсерверима наредбу за ажурирање копије документа. Да се на серверу не би стално креирале нити које служе за слање захтева за ажурирањем докумената, на централном серверу постоји група нити која обавља овај посао. Број нити је параметар који се задаје приликом рада серверске апликације. Централни сервер не шаље захтеве за ажурирањем директно нитима већ их убацује у јединствени бафер захтева за ажурирањем докумената. Нити узимају захтев по захтев из бафера и прослеђују га одговарајућем подсерверу у редоследу у коме су и пристизали. Информација о успешном ажурирању свих докумената се записује у лог.

Уколико је подсервер недоступан у моменту када се упућује захтев за ажурирањем покушава се поново након m/3 секунди. Уколико ажурирање свих подсервера на којима се налази копија документа не успе у интервалу од m секунди, тај захтев се проглашава необрађеним и то се уписује у лог. Притом се у лог уписују и подсервери на којима није успело ажурирање, али се поступак слања наставља све док се подсервер сматра исправним. Подсервер на коме од тренутка покретања није успело k захтева за ажурирањем проглашава се за неисправан и чека се његово поновно покретање. Уколико се неки подсервер не одазове извесно време (x секунди) такође се проглашава за неисправан. Неисправном подсерверу се не шаљу захтеви за ажурирањем, нити се одговара на захтеве за страницама док се он не јави централном серверу и потврди да је поновно покренут. Приликом покретања, подсервер брише све копије довучених докумената.

Водити рачуна да подсервери и сервер могу да приступају само документима који се налазе почев од специфицираног директоријума, односно да није дозвољено да се приступа документима који су изван

специфицираног директоријума. Специфицирани директоријум може да има структуру произвољног нивоа дубине. Уколико тражени документ не постоји на подсерверима и серверу, кориснику треба вратити страницу/документ са стандардном поруком да се јавила грешка 404. Ову поруку подсервер треба да врати клијенту и у случају да је централни сервер недоступан, а да не поседује копију траженог документа.

Клијенти имају могућност додавања докумената и мењања докумената и њиховог садржаја. Све измене треба да буду видљиве свим клијентима који су затражили документ или га тренутно мењају. Дозвољено је да више клијената истовремено мења документ, при чему клијенти могу да раде неометано (одзив мора бити задовољавајући). Уколико подсервер у међувремену постане недоступан, треба урадити балансирање свих клијената на све доступне подсервере (load balancing).

Проблем решити користећи мрежну комуникацију у програмском језику Јава. За сваки од ова три типа рачунара треба да постоји одговарајући графички кориснички интерфејс (GUI треба да буде развијен користећи Java SWING компоненте или JavaFX). Графички кориснички интерфејс мора бити интуитиван. За сваки од могућих случајева коришћења система поруке и обавештења морају бити прецизно приказани. Такође, потребно је да се кроз графички кориснички интерфејс прикажу детаљни логови, како за централни сервер, тако и за подсервере. Сервери треба да имају могућност покретања и без корисничког интерфејса. Такође, IP адресе и портови потребни за функционисање програма (као и остали подаци који се могу динамички мењати) морају се уносити преко графичког интерфејса (не смеју бити унапред дефинисани у изворном коду).

Предлог решења:

За решење овог проблема потребно је направити 3 различита пројекта. Пројекат за серверску апликацију, пројекат за подсерверску апликацију као и за пројекат за клијенте. У наставку ће бити дато објашњење како је реализована свака од њих и њихова комуникација. Упутство за употребу следи касније, као и опис рада у нерегуларним ситуацијама. За мрежну комуникацију коришћени су сокети (Sockets).

Серверска апликација:

Имплементирано је два типа нити, нит за комуникацију са подсервером и нити које управљају захтевима за сихнронизацију докумената. Нит која управља захтевима управља и захтевима од клијената (више о овоме у даљем тексту). Главни сервер (у даљем тексту сервер) , који је по претпоставци само један, код себе чува листу активних подсервера, као и који документ се налази на ком подсерверу. Нити које управљају захтевима периодично пролазе кроз све парове документ-подсервер и за сваки пар убацују нови захтев у бафер захтева. Бафер захтева је имплементиран као конкурентна структура LinkedBlockingQueue iz Java.Concurrent пакета. Нит за комуникацију периодично узима захтеве из бафера и комуницира са подсервером чија је IP адреса дата у захтеву. У зависности од тога чији документ је касније модификован, сервер ће или узимати документ са подсервера, или га слати на исти.

Подсерверска апликација:

На подсерверској апликацији је такође имплементирано два типа нити, нит за комуникацију са клијентима или сервером и нити које управљају захтевима за сихнронизацију докумената. Нити које управљају захтевима периодично узимају захтев који је пристигао преко мреже, стављају га у бафер захтева, заједно са информацијом да ли је захтев стигао од клијента или од сервера. Бафер захтева је имплементиран као конкурентна структура PriorityBlockingQueue iz Java.Concurrent пакета, који приоритизира захтеве од сервера при извлачењу из бафера. Нит за комуникацију периодично узима захтеве из бафера и комуницира са сервером или клијентом чија је IP адреса дата у захтеву. У зависности од тога чији документ је касније модификован, сервер ће или узимати документ са сервера/клијента, или га слати на исти.

Клијентска апликација:

На клијентској апликацији постоји само тип нити која комуницира са сервером или подсервером. Клијентска нит ће на почетку тражити од сервера да јој пошаље IP адресу једног активног сервера, а затим ће клијент покушати да успостави конекцију са подсервером тако што ће слати захтев за синхронизацију и чекати на баш тај захтев на подсерверу. У зависности од тога чији документ је касније модификован, подсервер ће или узимати документ са клијента, или га слати на исти, уколико ниједан није био модификован, у лог-овима ће писати да је документ синхронизован. Пошто клијент тренутно ради на неком текстуалном фајлу, уколико он притиска тастатуру док користи програм, време модификације тог фајла ће се мењати и оно што је укуцано ново, биће синхронизовано, прво на подсервер, а затим и на сервер и остале подсервере који имају тај фајл.

Рад програма у непредвиђеним ситуацијама:

Као и свака дистрибуирана апликација, и ова апликација је подложна разним непредвиђеним ситуацијама. Приликом решавања проблема, успешно је покривено неколико случајева непредвиђених ситуација, наводимо неке од њих:

1. Ситуација у којој серверска апликација престане са радом при комуникацији са подсервером:
   Приликом ове ситуације програм покушава да се прилагоди, тако што уколико је у тренутку престанка рада сервера сервер слао фајл на подсервер, подсервер је пре него што је потврдио слање фајла, сачувао резервну копију фајла ког је сервер требао да пошаље (тиме и синхронизовати тај фајл), па ће покрварену верзију обрисати и резервну преименовати.

2. Ситуација у којој клијентска апликација престане са радом при комуникацији са подсервером:
   Приликом ове ситуације програм покушава да се прилагоди, тако што уколико је у тренутку престанка рада клијентске апликације клијент слао фајл на подсервер, подсервер је пре него што је потврдио слање фајла, сачувао резервну копију фајла ког је подсервер требао да пошаље (тиме и синхронизовати тај фајл), па ће покрварену верзију обрисати и резервну преименовати.

3. Ситуација у којој серверска апликација престане са радом при комуникацији са клијентском приликом тражења IP адресе подсервера:
   Уколико се сервер прекине при комуникацији са клијентском приликом тражења IP адресе подсервера, онда се сервер сматра неисправним и клијенту се исписује порука да сервер не ради, и да мора поново да покрене апликацију.

4. Ситуација у којој подсерверска апликација престане са радом при комуникацији са клијентском:
   Приликом ове ситуације програм покушава да се прилагоди, тако што уколико је у тренутку престанка рада подсерверске апликације подсервер слао фајл на клијента, клијент је пре него што је потврдио слање фајла, сачувао резервну копију фајла ког је подсервер требао да пошаље (тиме и синхронизовати тај фајл), па ће покрварену верзију обрисати и резервну преименовати. Такође уколико клијент пошаље N захтева на подсервер и они не успеју да дођу до подсервера, **клијент ће се јавити серверу** и тражити нови подсервер од њега, тиме испуњавајући **load balancing** својство а сервер ће прогласити тај подсервер неисправним, и више неће слати захтеве том подсерверу (уколико и сервер не одговара приликом ове ситуације, кориснику ће се исписати порука да је сервер недоступан и да мора поново да покрене апликацију).

Неке ситуације **нису имплементиране**, али постоји могућност да се накнадно додају и побољшају рад програма.

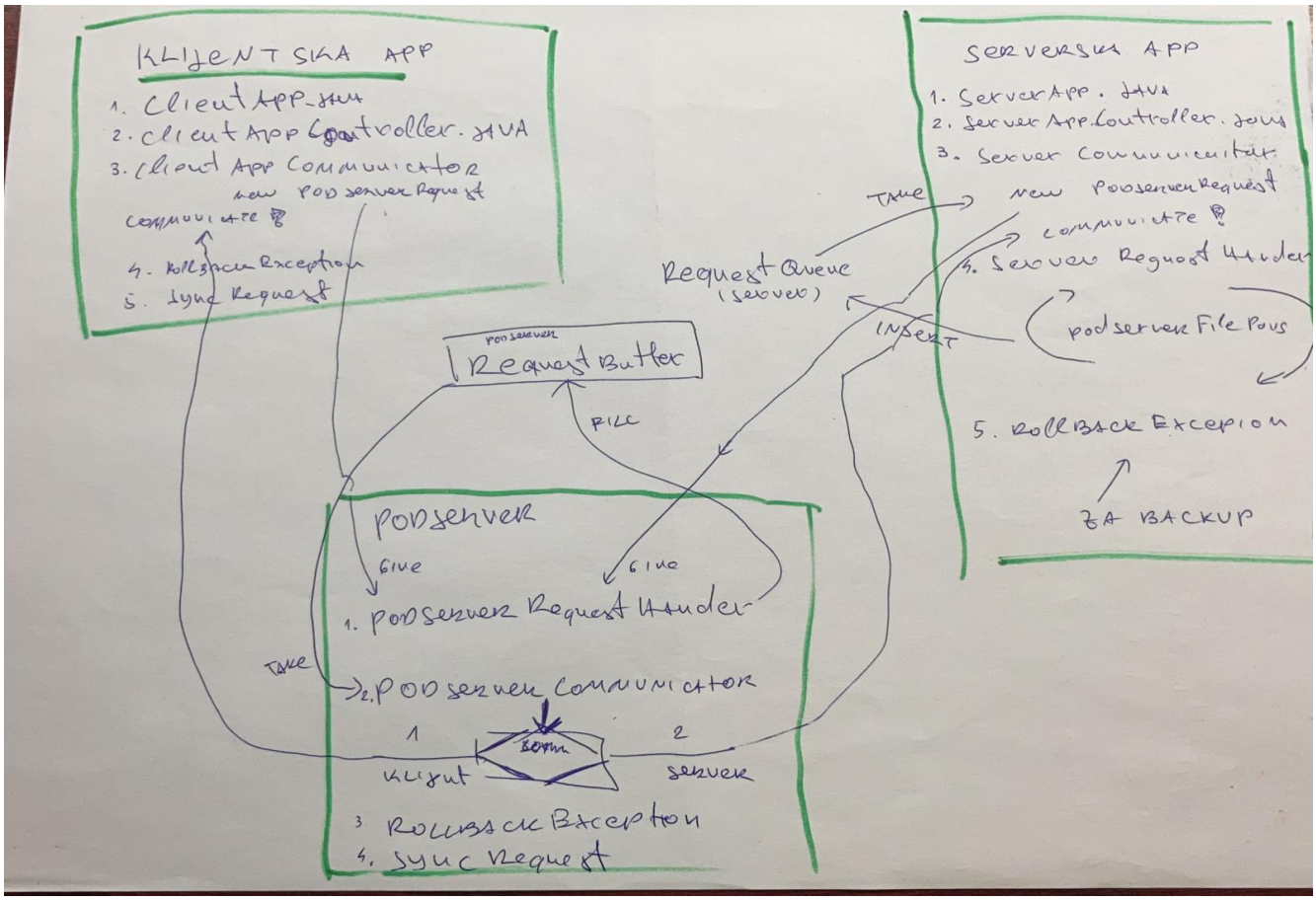
- Уколико подсервер престане са радом, и једном је избачен са листе подсервера, накнадно се може само додати у листу подсервера, али му се не могу вратити оригинални клијенти, који су касније пребачени на неки други подсервер.

- Уколико сервер престане са радом, а подсервери раде, изубиће се синхронизација истих фајлова на различитим подсерверима, међутим ово је **делимично решено статички**, тако што сервер када га клијент контактира да тражи подсервер за одређени фајл, сервер приоритизује онај подсервер који га већ поседује, тражи пар подсервер – фајл.

- Уколико сервер престане са радом, а подсервери раде, није могуће да сервер поново комуницира са истим подсерверима, зато што је листа подсервера и парова подсервер – фајл реализована као променљива, није нигде трајно записана

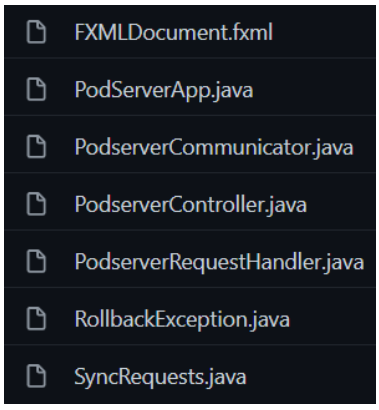
Опис спецификације (шема)

На следећим сликама дата је руком нацртана шема која сликовито описује рад програма, који је већ описан у предлогу решења.

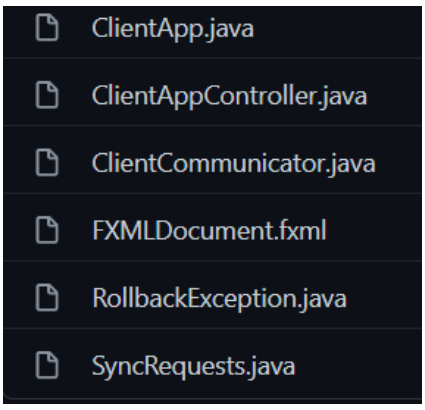


Слика 1. Шема

Слика 2. Подсерверска апликација:



Слика 3. Серверска апликација:



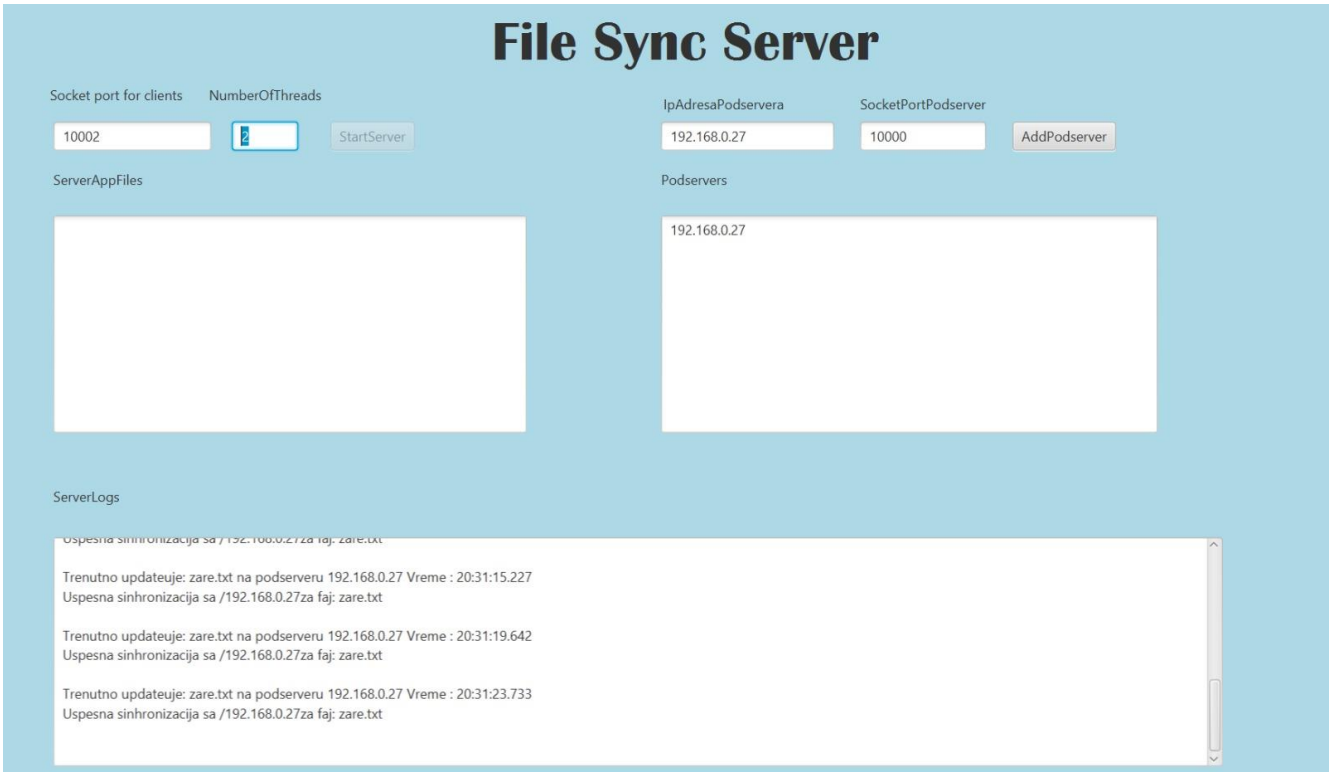
Слика 4. Клијентска апликација:

Упутство за употребу:

Упутство за употребу ће бити подељено у три дела, јер постоје три апликације.

Препоручује се следећа секвенца стартовања програма. Серверска апликација, клијентска апликација, подсерверска апликација због најмањег чекања и најмање шансе да се нека комппонента прогласи неисправном.

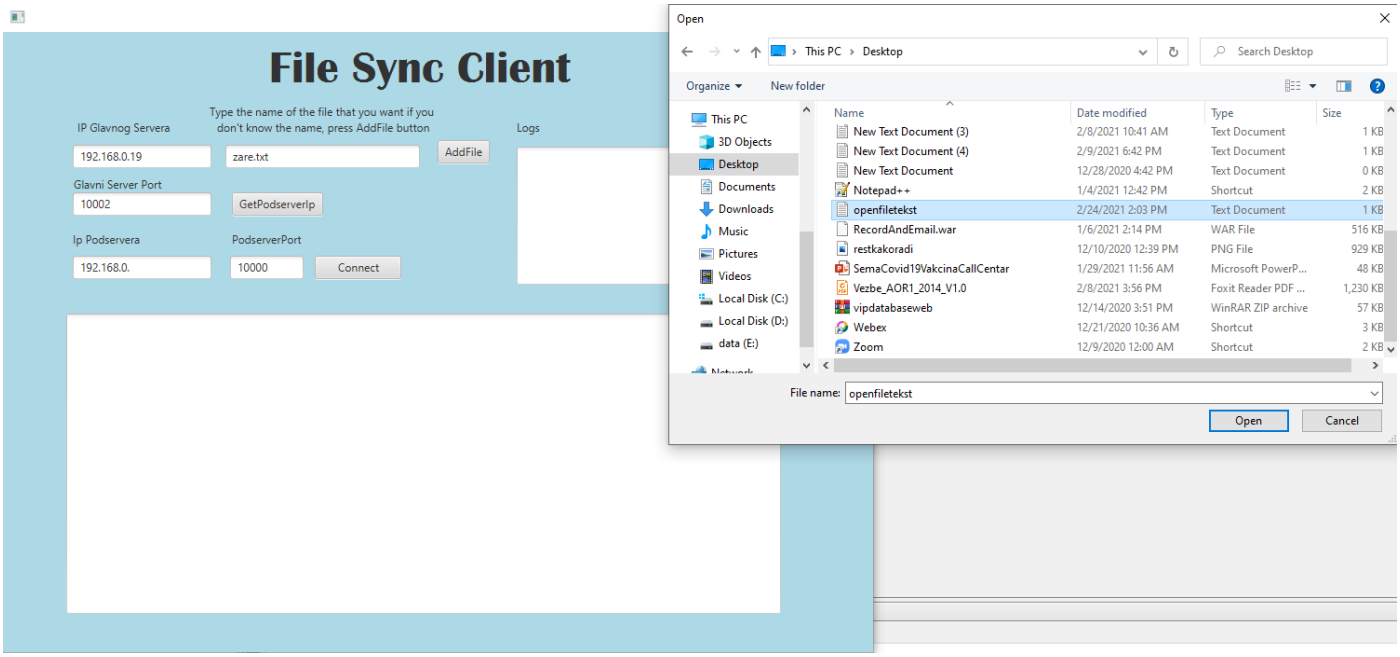
Серверска апликација:



Слика 5. Серверска апликација при раду

Након покретања ServerApp.jar отвориће се серверска апликација. Како би корисник покренуо сервер, мора прво унети IP адресу подсервера и додати све познате подсервере, тако што ће притиснути дугме AddPodserver. Затим је потребно да корисник унесе порт на ком сервер и клијенти комуницирају и број нити који ради на слању захтева за синхронизацију и притисне дугме StartServer. На слици 1 је приказана серверска апликација при раду.

Клијентска апликација:

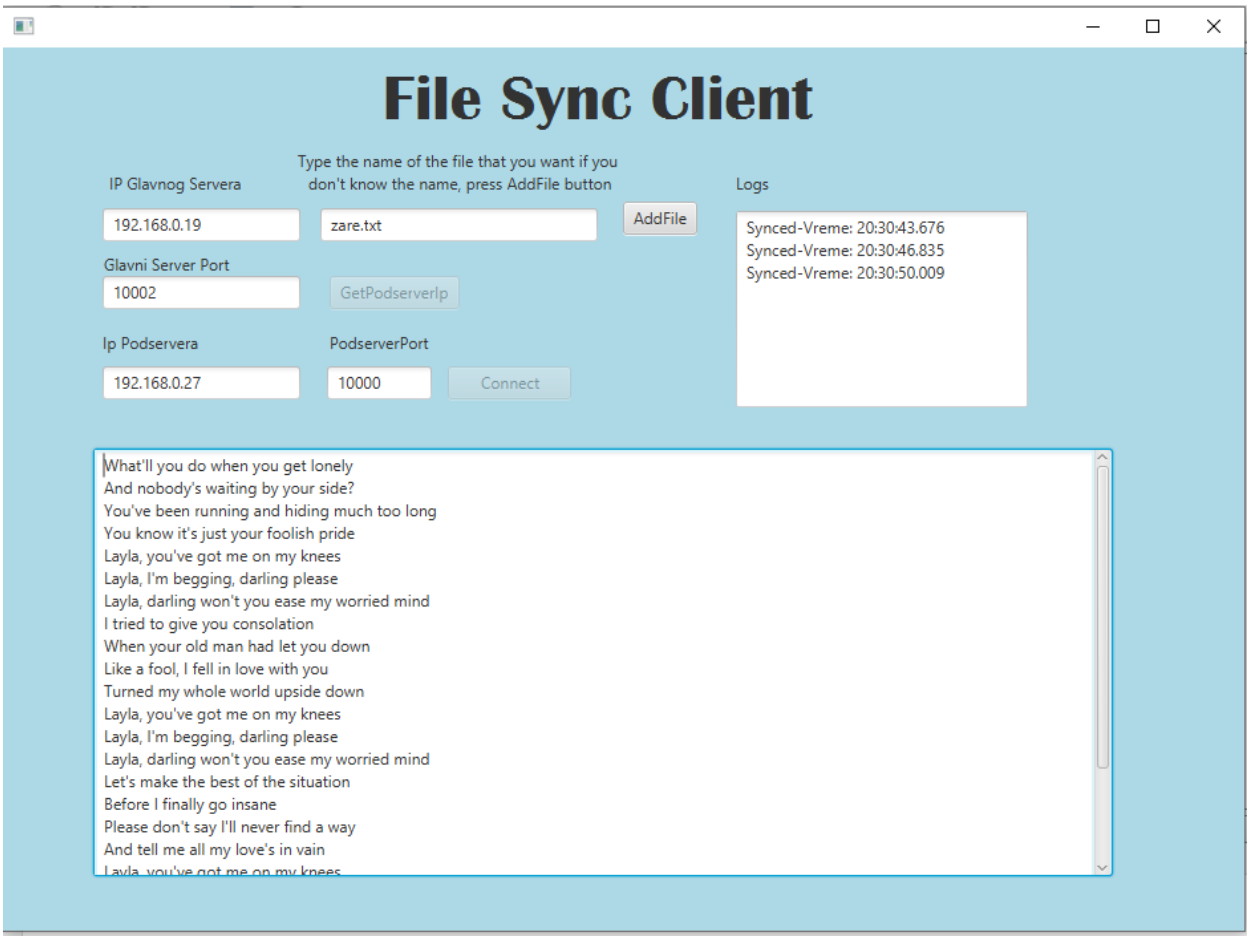


Слика 6. Клијентска апликација на почетку

Након покретања ClientApp.jar отвориће се клијентска апликација. Како би корисник учитао фајл, треба притиснути дугме AddFile а затим одабрати фајл који жели. Ограничење овог програма су само фајлови који су текстуалног типа, међутим уколико не би морало да се приказује на апликацији, овај програм би могао да шаље фајлове било ког типа. Затим је потребно уписати IP адресу подсервера, и написати порт са којим он комуницира са клијентима и притиснути дугме GetPodserverIp са којим корисник од главног сервера добија адресу подсервера на који треба да се накачи. Затим ће се TextBox Ip Podservera променити на адресу коју је сервер вратио.

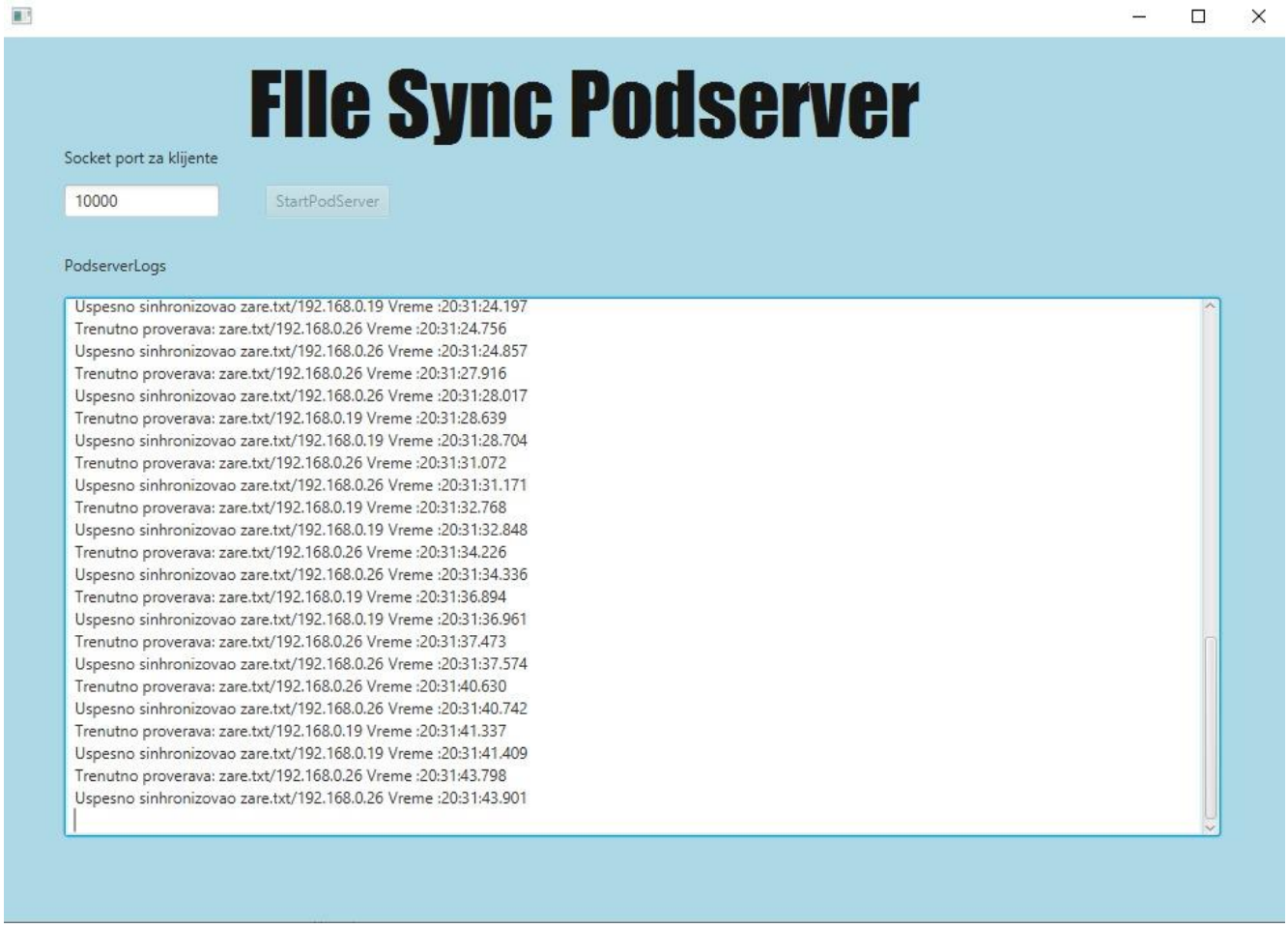
Када корисник притисне дугме Connect, клијент ће учитати текст, и моћи ће да га мења. Видети слику 2. на следећој страни.

Такође ће у делу Logs бити исписани логови, који приказују време када се синхронизација десила, као и уколико дође до непредвиђених ситуација.



Слика 7. Клијентска апликација при раду

Подсерверска апликација:



Слика 8. Подсервер при раду

Након покретања PodServerApp.jar отвориће се подсерверска апликација. Подсерверска апликација се покреће дугметом Start, док се пре тога мора унети порт са којим подсервер комуницира са клијентима (сервером и клијентима). На слици 4. је приказана подсерверска апликација при раду.