# תיאור המערכת

## ישויות במערכת

### הגדרות

* לקוח – Client
* שרות חיפוש – FlightSearchServer
* מוכרן – AirlineServer
* שאילתה – בקשת query שהגיעה מלקוח
* קונקשן - טיסה שניה המתאימה לדרישות התרגיל שמוחזרת בשאילתה

### תיאור

המערכת מורכבת מ-3 ישויות מרכזיות בדומה לתרגיל הקודם:

1. הלקוח שהינו צרכן המידע במערכת והמקור לגירויים.
2. שרות החיפוש (FlightSearchServer) שמפנה את הבקשות ליעד המתאים
3. שרות Alliance המאגד מספר מוכרני טיסות לישות אחת. ה-Alliance מספק שרות פונקציונלי עבור הלקוח ע"י מציאת מסלול מתאים ממוצא כלשהו ליעד כלשהו דרך לכל היותר יעד שלישי (הטיסה השניה הינה טיסת קונקשן) מכל המוכרנים השייכים ל-Alliance ושירות פונקציונלי ביחס לשרתי מוכרנים שהם חלק מה- Alliance והוא שרות רפליקציה העמיד בפני n-1 נפילות כאשר n הינו מספר המוכרנים המשתתפים. שרות הרפליקציה מאפשר נגישות למוכרנים אשר שרתיהם נפלו דרך שרתים של מוכרנים אחרים ובנוסף מאשר לבצע Load balancing של כמות המוכרנים למכונה (נשים לב כי פרמטר ה-Load balancing אינו לפי עומס בקשות אלא לפי מספר מוכרנים למכונה).  
   כל Alliance חושף עצמו לשרות החיפוש ע"י delegate, לכל Alliance יש delegate יחיד.

## תיאור עץ ה-ZooKeeper

בכדי לזהות שינויים במערכת באופן קונסיסטנטי דרוש מנגנון קונצנזוס עם יכולת MEMBERSHIP. בתרגיל השתמשנו ב-ZooKeeper למטרה זו שכן המערכת מספקת ספריה מונחת אירועים.



## תיאור מודולרי של התכנית

### לקוח

מאפשר ביצוע שאילתות לטיסות עפ"י מועד יציאה, יעד, מוצא ופרמטר אופציונלי הבוחר את המוכרן אליה יש להפנות את הבקשה. אם הבקשה לא מפנת למכרן מסויים היא תשלח לכל המוכרנים ולכל ה Alliance הרשומים למערכת החיפוש.

הלקוח יציג את תוצאות החיפוש ממויינות עפ"י מחיר

### שרת FlightSearchServer

שרת זה מהווה אגרגטור בקשות של לקוחות ומעביר אותם ל-Delegates הרשומים אצלו. ה-Delegates עונים לו ומחזירים רשימות של טיסות המתאימות לקריטריון והוא מאחד, ממיין ומחזיר את הרשימות ללקוח תוך כדי הוספת השייכות של התוצאה (כלומר, לאיזה Alliance התוצאה שייכת). כאשר Delegate נופל תפקיד ה-Delegate החדש להרשם מחדש.

### שרת AirlineServer

#### מודולים במערכת:

##### מודול TreeView

תפקידו של מודול זה הוא לנהל את ה-Group Membership View הלוקאלי לקרא לפוקנציות callback מתאימות כאשר יש שינוי במערכת (הצטרפות או נפילת מכונה). המודול מורכב ממספר תת מודולים בהם ה-TreeView עצמו שהינו מסד נתונים מקומי המשקף את מצב המערכת ומודול AirlineReplicationModule שמנהל את החיבור מול שרת ה ZOOKEEPER ובעצם מתחזק את ה TREEVIEW מאירועים שהוא מקבל כ- LOCAL GROUP VIEW.

כמו כן, ה AirlineReplicationModule גם אחראי לקרא לפעולות התחזוקה השוטפות כאשר קורים אירועים מעניים במערכת (נפילות והצטרפויות של שרתים).

להלן סכימה המתארת את עליית המודול:



##### מודול ראשי

###### מודול AirSellerRegistration

מודול זה אחראי על ביצוע אלגוריתם האיזון ובקרת הנפילות

להכניס עוד מידע....

###### מודול ה-Cache

מודול זה אחראי על מנגנון המטמון

להכניס עוד מידע ....

## ממשקים – API

### ממשק שרת חיפוש

#### בפני לקוחות

ממשק מסוג **REST** החושף את הפונקציונליות הבאה:

* **/Services/FlightsSearch/flight?src={src}&dst={dst}&date={date}&servers={servers}**
* פונקציונליות לקוח
* סוג GET
* *ערך חזרה:* מחזירה רשימה של תוצאות
* *תיאור:* הפונקציה שולחת שאילתה לשרת החיפוש. הפונקציה תחזיר רשימה של טיסות (כולל טיסות עם קונקשן) ממקור src ליעד dst בתאריך date מהשרת servers.

#### בפני שרתי מוכרנים ו-Alliance delegates

ממשק מסוג **SOAP** החושף את הפונקציונליות הבאה:

* **Services/FlightsSearchReg/Register/{clustername}**
* Uri – Cluster uri
* סוג PUT

שרת delegate של Alliance מסוים או מוכרן כלשהו המעוניין לקבל בקשות חיפוש מהשרת מפעיל בקשה זו.

* **Services/FlightsSearchReg/Unregister**
* אין קלט
* *תיאור:* הפונקציה תשתמש בכתובת המקור ממנה הבקשה הגיעה כדי להסיר את ה Alliance מרשימת השרתים שאליהם שולחים חיפושים.

### ממשק שרת Airline server

#### בפני שרתי חיפוש ומוכרנים המתפקדים כ-Delegates של Alliance:

ממשק מסוג **SOAP** החושף את הפונקציונליות הבאה:

* **/Services/SellerService/getTrips**
* Src – מקור הטיסה (String)
* Dst – יעד הטיסה (String)
* Date – תאריך הטיסה (DateTime)
* Sellers – אופציונלי, אם הלקוח פירט רשימת מוכרנים מסוימת (List<String>)
* *ערך חזרה:* List<Trip> כאשר Trip הינו אובייקט המכיל את כל המידע שנדרש להצגה אצל הלקוח.
* *תיאור:* השרת המתפקד כdelegate חושף שירות זה המאפשר ביצוע שאילתה על ה-Alliance. אם הוגדרו מוכרנים וה-delegate אינו מכיר אותם הוא יתעלם מהבקשה ויחזיר רשימה ריקה.

#### ממשק תחזוקה פנימי בין שרתי מוכרנים

ממשק מסוג **SOAP** החושף את הפונקציונליות הבאה:

* **/Services/IntraClusterService/sendPrimarySeller**
* sellerName – שם המוכרן (String)
* *ערך חזרה:* מסד נתונים המייצג את כל המידע אותו המוכרן מחזיק.
* *תיאור:* פונקציה זו מבקשת מהמכונה המריצה את השרות את מוכרן ראשי אותו היא מחזיקה (יתכן שבעת נפילות מכונה מחזיקה מספר מוכרנים כראשיים – כלומר פעילים), פונקציה זו משמשת מוכרנים אחרים בעת ביצוע תהליך Load Balancing בעת נפילת או הצטרפות מכונה חדשה. ערך החזרה עלול להיות גדול מאוד שכן הוא מייצג את מסד הנתונים כולו.
* **/Services/IntraClusterService/sendBackupSeller**
* sellerName – שם המוכרן (String)
* *ערך חזרה:* מסד נתונים המייצג את כל המידע אותו המוכרן מחזיק.
* *תיאור:* פונקציה זו מבקשת מהמכונה המריצה את השרות את מוכרן גיבוי אותו היא מחזיקה (יתכן שבעת נפילות מכונה מחזיקה מספר מוכרנים כגיבוי), פונקציה זו משמשת מוכרנים אחרים בעת ביצוע תהליך Load Balancing בעת נפילת או הצטרפות מכונה חדשה. ערך החזרה עלול להיות גדול מאוד שכן הוא מייצג את מסד הנתונים כולו.
* **/Services/IntraClusterService/getRelevantFlightsBySrc**
* Src – נקודת היציאה של הטיסה (String).
* Date – תאריך היציאה של הטיסה (DateTime)
* *ערך חזרה:* רשימת טיסות List<Flights> שיוצאות מ-src בתאריך Date.
* *תיאור:* פונקציה זו נקראת ע"י ה-Delegate של ה-Alliance במטרה לנסות להרכיב טיסות היוצאות בתאריך ה-Date (הרכבת קונקשן).
* **/Services/IntraClusterService/getRelevantFlightsByDst**
* Dst – יעד הטיסה (String)
* Date – תאריך היציאה של הטיסה (DateTime)
* *ערך חזרה:* רשימת טיסות List<Flights> המגיעות ל-Dst ויוצאות בתאריך Date.
* תיאור: פונקציה זו נקראת ע"י ה-Delegate של ה-Alliance במטרה לנסות להרכיב טיסות היוצאות בתאריך ה-Date (הרכבת קונקשן).

שאלה יבשה 5

נציע מימוש לפעולות reserve ו-cancel אך לפני כן נניח כי עבור טיסות עם קונקשן הלקוח ישלח בשאילתה את מספרי הטיסות (מקור-קונקשן, קונקשן-יעד).

שיטה 1:

אין שינוי בפרדיגמת התקשורת בין הלקוח לבין שרת החיפוש ובין שרת החיפוש לבין הdelegate של ה-cluster (הלקוח ישלח את הבקשה ושאר הישויות בדרך ישמשו כ"נתב מסובך").

מאחר ושרת הdelegate הוא נקודת הכניסה והיחידה היציאה ל-cluster ומאחר ושרתים ב-cluster אינם משפיעים זה על זה כאשר מקבלים בקשת cancel/reserve מספיק לנעול את זוג המוכרנים לבקשות נוספות מסוג זה. הנעילה תתבצע ע"י הגדרת watch בכל שרת על מסלול אפשרי למנעול עבור מוכרן primary אותו הוא מחזיק (znode מסוג persistent). מוכרן נעול יכול לענות לשאילתות Query אך אינו עונה לשאילתות cancel/reserve.

כאשר ה-delegate יקבל הודעה מסוג cancel/reserve הוא יבדוק האם המוכרנים אליהם הבקשה מיועדת

מכאן ניתן לפתור את הבעיה בשני אופנים שונים כאשר לכל