תיאור המערכת

ישויות במערכת

הגדרות

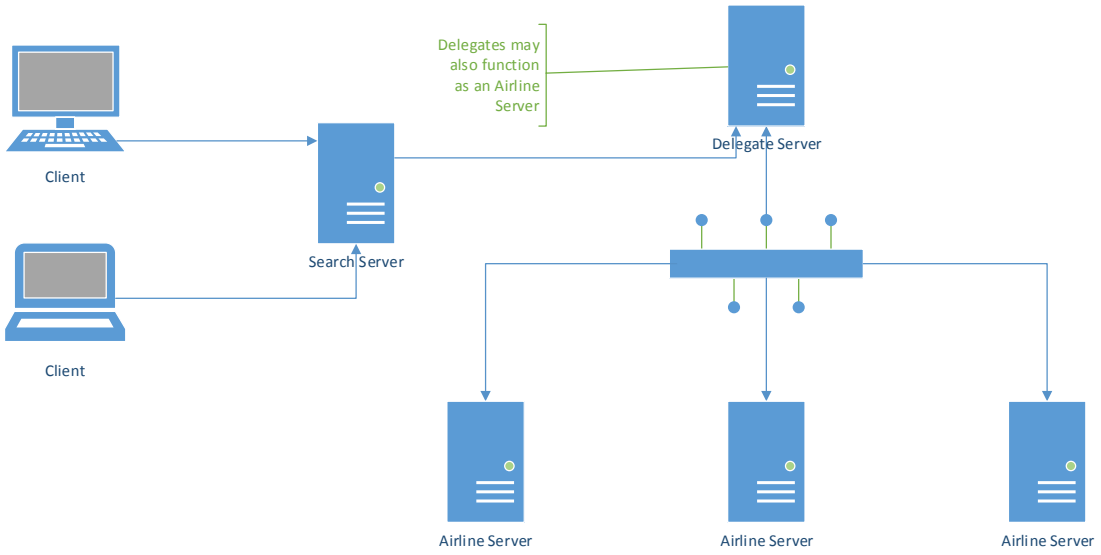
- לקוח – Client
- שרות חיפוש – FlightSearchServer
- מוכרן – AirlineServer
- שאילתה – בקשת query שהגיעה מלקוח
- קונקשן - טיסה שניה המתאימה לדרישות התרגיל שמוחזרת בשאילתה

תיאור

המערכת מורכבת מ-3 ישויות מרכזיות בדומה לתרגיל הקודם:

1. הלקוח שהינו צרכן המידע במערכת והמקור לגירויים.
2. שרת 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 השנייה 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.



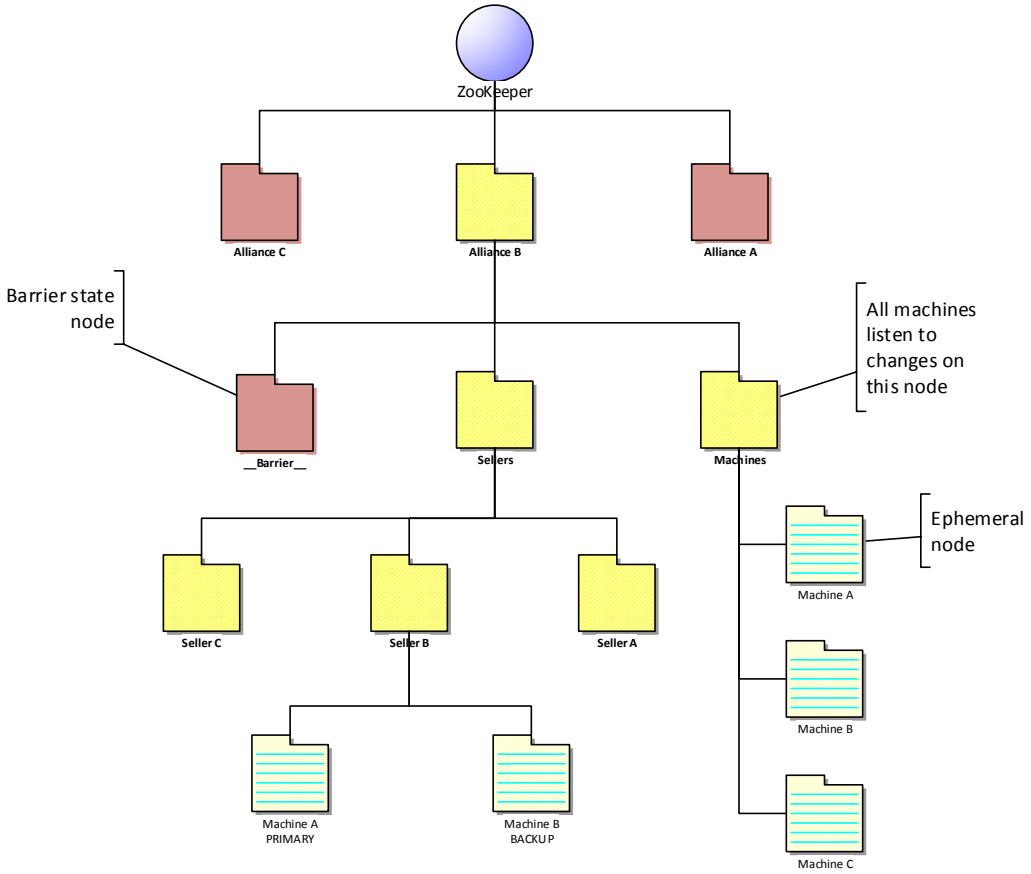
כדי להתמודד עם נפילות בחרנו

הממשק מול -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 וכן בנינו תשתית לניהול עץ לוקאלי המתעדכן בשינויים (Diff Merging) שמנוהלת ע"י מנגנון הרפליקציה.

המערכת מתוכננת כך שהיא מפרידה בין המידע הלוגי (מוכרן) לבית היחידה החישובית המריצה אותו (מכונה). באופן זה ניתן לבצע העברה יעילה של יחידות לוגיות בין מכונות שונות ולבצע ניתור נפילות ובחירת Delegate.

להלן תרשים המציג את היררכית העץ:



הצמתים הצהובים הם צמתים שכל ישות ב-Alliance מכירה (בהתאמה ל-Alliance בו היא נמצאת).

עבור כל Alliance קיים תת-עץ תחת שורש עץ ה ZooKeeper, (מעתה עץ ה-Alliance). תחת עץ זה רשומים כל המכונות והמוכרנים אותם ה-Alliance.

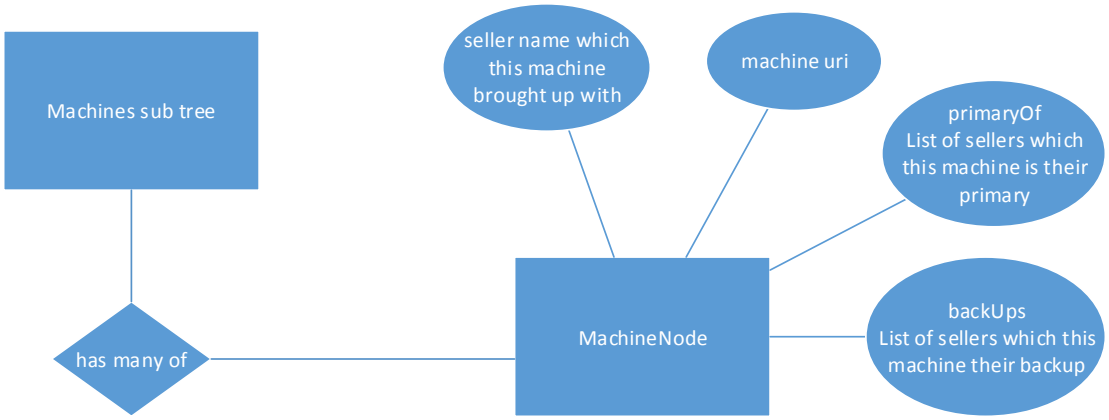
עבור כל מכונה שמצטרפת למערכת יוצרים עלה בענף המכונות (alliance/machines/) מסוג ephemeral כך שכל מי שנרשם לאירועי עדכונים על ענף זה יקבל אירוע עדכון במידה ומכונה הרשומה בענף זה כשלה כמו כן, אם לא קיים ענף עבור המוכרן איתו היא עלתה היא יוצרת עבורו ענף ורושמת עצמה כבן מסוג PRIMARY שלו.

לכל מוכרן המצטרף למערכת יש לכל היותר 2 בנים המסמלים את המכונות שמחזיקות את המוכרן הזה (אחת כגיבוי ואחת ראשית). בנים אלו הם מסוג Ephemeral כך שבעת כשל

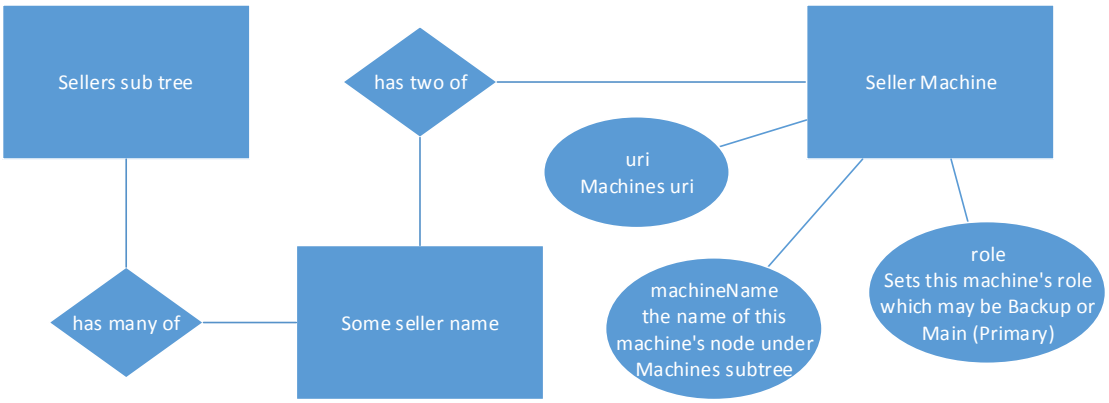
הוא ימחק מהמוכרן וה-Delegate של ה-Alliance לא יראה אותו יותר ולכן ייגש ישירות לשרת חי המחזיק העתק שלו.

כל עלה בענף המכונות מחזיק זוג רשימות המתארות את המוכרנים שהוא "מחזיק" (יכל לתת שרות עבורם), רשימה אחת עבור המוכרנים שהוא משרת כרגע באופן פעיל (ה-delegate במערכת ניגש אליו עבור שאילתות) ורשימה נוספת עבור מוכרנים שהוא מחזיק כגיבוי (במידה ולמוכרן לא קיימת מכונה המוגדרת כראשית יש לגשת למכונת הגיבוי כדי לקבל שירות).

להלן תיאור המידע המוחזק ע"י עלים בענף המכונות



להלן תיאור המידע המוחזק ע"י עלים של מוכרן בענף המוכרנים



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 ממוינות עפ"י מחיר שפורט במסמך הדרישות של התרגיל.

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 להירשם מחדש.

אם שרת החיפוש פנה ל- Alliance בזמן שהמערכת מבצעת פעולת איזון אזי ייתכן כי המערכת תחזיר תשובה לא מלאה (יורחב בהסבר על שרת AirlineServer).

שרת AirlineServer

מודולים במערכת:

מודול TreeView

תפקידו של מודול זה הוא לנהל את ה-Group Membership View הלוקאלי לקרא לפונקציות 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, מאזין לשינויים בענף המכונות של ה-Alliance אליו הוא שייך ומבצע פעולות Merge על ה TREEVIEW מאירועים שהוא מקבל, באופן זה ה-View הלוקאלי תואם תמיד לזה של ה-ZooKeeper.

AirlineReplication חושף ממשק אירועים (Callbacks) וממשק פונקציונלי שבעזרתו ניתן לקבל תמונת מצב לוקאלית של ה-Alliance ולהשפיע על מצב המערכת ועל כניסות בעץ ה-ZooKeeper **השייכות למכונה שרצה**, כלומר, מכונה שרצה אינה יכולה להשפיע על מבני נתונים שאינם שייכים באופן מפורש אליה. כשמתבצעת בקשת שינוי למבנה נתונים כזה השינוי יתבצע באופן לוקאלי בלבד ולא יישלח ל-ZooKeeper.

מימוש זה מאפשר להפריד באופן מוחלט את אלגוריתם ה- Load Balancing וטיפול בנפילות (שהם בעצם הינו הך מאחר וכשמטפלים בנפילות עושים זאת עם שיקולי עומס).

כאשר המערכת מתעוררת היא מבצעת אתחול לאובייקט הרפליקציה (AirlineReplicationModule). כאשר אובייקט הרפליקציה מאותחל הוא מבצע אנומרציה לענפים בעץ הרלוונטיים אליו, מוסיף את המוכרן השייך לו באם לא קיים עבורו ענף, נרשם לאירועי שינוי בעץ ומצטרף בעצמו לענף המכונות והמוכרנים של אותו Alliance אליו הוא שייך.

הצטרפות זו גוררת אירוע שינוי בעץ, דבר אשר מפעיל את פונקצית הטיפול בהצטרפות מכונה חדשה. פונקציית ההצטרפות בוחנת את השינויים בעץ הלוקאלי, מבצעת עליו Diff Merge, קוראת לפונקציית callback ומעבירה לה Snapshot של המערכת ופרטי המכונה שהצטרפה ובסיום הקריאה היא נרשמת מחדש לשינויים בענף המוכרנים בעץ.

כאשר מכונה נופלת מופעלת פונקציית העזיבה אשר בוחנת את השינויים בעץ הלוקאלי, מבצעת עליו Diff Merge, קוראת לפונקציית callback ומעבירה לה Snapshot של המערכת ופרטי המכונה שנפלה ובסיום הקריאה היא נרשמת מחדש לשינויים בענף המוכרנים בעץ.

בשני המקרים, הדרך של ה-Callback להשפיע על מצב המערכת הוא בעזרת פונקציה בשם updateMachineData אשר מקבלת פרמוטציה על ה-Snapshot אותו ה-Callback קיבלה.

הפונקציה updateMachineData בוחנת את ההבדלים בין המידע שהיא קיבלה לבין העץ הנוכחי ומבצעת עדכונים **רק היכן שיש שינוי** (מטעמי אופטימיזציה) ומעדכנת בעץ ה-ZooKeeper רק את הרשומות בהן המכונה הופיעה ושוב, **רק היכן שיש שינוי** כדי לחסוך בקריאות מיותרות.

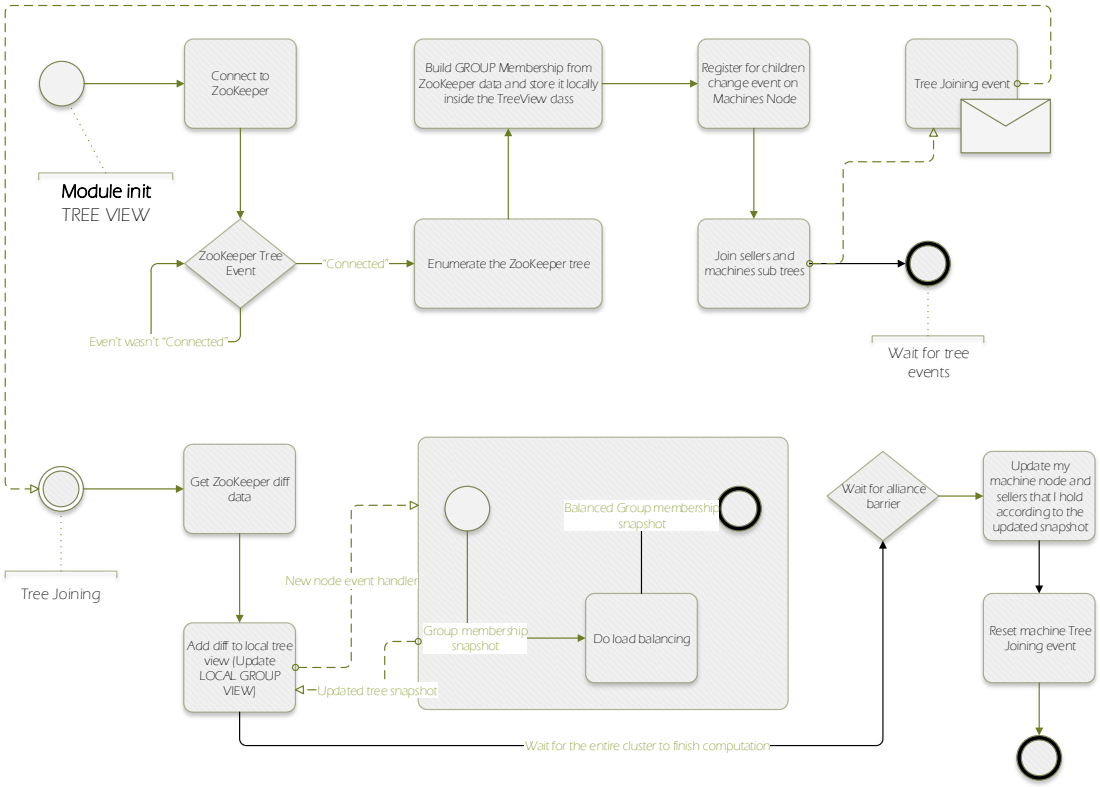
המידע שעלול להשתנות ב-ZooKeeper בעקבות הקריאה לפונקציה זו הוא:

1. הסרת עלה השייך למכונה בענף המוכרנים (כי אלגוריתם הניתוב החליט שהמכונה כבר אינה גיבוי\ראשית).
2. שינוי רשומת המכונה בענף המוכרנים (המכונה עברה בין המצבים BACKUP ו-PRIMARY).
3. צירוף המכונה כעלה חדש במוכרן במידה ולא היה קיים לפי הפרמטרים שמתאימים לנתונים ב-Snapshot החדש שקיבלה (Main/Primary).
4. שינוי רשומת המכונה בענף המכונות.

נשים לב כי ל-ZooKeeper יש 2 תפקידים עיקריים:

1. החזקת Group Membership שבעזרתו מכונות חדשות יכולות תמונת מצב של המערכת ולבנות Snapshot.
2. מערכת גירוי המספקת אירועים ומידע בהצטרפות\עזיבה של תחנות.

להלן סכמה המתארת את עליית המודול ומודל לטיפול באירוע עליה של מכונה חדשה:



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 ....

טיפול בנפילות

בלה בלה

טיפול בהצטרפות

בלה בלה

גדשדשג

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}**
  - Cluster uri – 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).

<u>שיטה 1:</u>

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