המכללה האקדמית להנדסה
אורט בראודה
ORT BRAUDE COLLEGE

Smart Courier: Server Side

Authors
Mor Gatya ID: 203781265, Barak Segal ID: 204336440

Supervisor
Dr. Elena Ravve

Abstract. *Shipping companies are growing up so fast nowadays, online shopping grows up exponentially. The couriers usually get a number of shipments that they should deliver, and they create their own track, trying to deliver the shipments in the specific time, while maximizing the economical savings as much as they can, by trying to take the shortest track between all the addresses that they should deliver packages to. A system that can help the shipping department to distribute the addresses among couriers, while the system is maximizing several factors and meeting strong constraints is strongly needed.*

Keywords: ABC algorithm, Server side, client side, Monte Carlo algorithm.

1. INTRODUCTION

Given a list of deliveries to be reached by several couriers is a prevalent problem. It includes distributing the deliveries' addresses between couriers while optimizing several factors, such as the total driving distance of each courier, distributing the load of urgent deliveries between couriers, and considering strong constraints. Our aim is to create a system, which handles all these features in the best possible way.

1.2 What Are We Going To Do?

The global aim of this multi-step project is to create a server and an Android-based application to manage a delivery system. The delivery system will distribute deliveries among couriers in an optimal way using multiple criteria. In this project, we are focusing on implementing a server side that will handle a database and will run the algorithm of the deliveries' distribution among couriers. In addition to that, we will implement a client that will interact with the server for maintenance (Adding or deleting information about couriers and deliveries), and for triggering the distribution algorithm. The team that worked on this multi-step project before us have already implemented an Android-based application that will serve the couriers and will interact with our server.

1.3 Why Is It Not Trivial?

From the algorithmically point of view, finding the optimal deliveries' distribution between couriers is a non-trivial challenge. In fact, different factors must be taken in account and multiple criteria must be optimized simultaneously. In addition, the algorithm should keep polynomial time complexity because the server can get a request to run the algorithm at any time of the day and by multiple clients. Moreover, from the software engineering point of view, the server side needs to interact with an Android-based application that another team has already developed, and the server side needs to interact with another client (maintenance client). Furthermore, the maintenance client should be easy to accesses from different computers and be user-friendly.

1.4 What Are The Difficulties Of The Project?

Our project is complicated. It will combine a lot of algorithms and platforms that we should work with. We might have some difficulties in our project such as:

1) Algorithms:

a. Deliveries' distribution should be in the complexity time of polynomial and will return the best optimized solution in high probability.
b. Different factors must be taken in account and multiple criteria must be optimized simultaneously.
c. Finding and calculating the drive distance of each delivery's address from each other to minimize the factor of distance.

2) Software Engineering**:**

a. To design our server, we must learn new technology for building a reliable and efficient server that will be able to respond fast to different requests from many users in parallel.
b. We will have to learn how to integrate the server side with the maintenance client, and with the Android-based application which other team built, in a fast and reliable way.
c. We must ensure that our server is secured from attackers that want to steal any information from the database.
d. The maintenance client side should be user-friendly and easy to access for multiple users around the world.
e. Access to the database should be fast and reliable, and database tables should be easy to expand.

1.5 How Do We Deal With These Difficulties?

To deal with the difficulties we have:

1) Algorithms:

a. We will develop an optimization algorithm that uses a Monte Carlo algorithm [12] that will have a polynomial time complexity. Moreover, the algorithm will return the best optimized solution at high probability P, and it will be optional for the user to improve this probability by increasing the iterations of the algorithm. Then the probability of the algorithm to return the best solution will be: $P_{new} = \frac{\ln\left(\frac{1}{P}\right)}{x}$. Where x is the number of times the algorithm was run.
b. Our algorithm will be based on the ABC algorithm who is a fast and reliable algorithm for optimizing several factors simultaneity, more details about this algorithm in section 2.
c. We will use google maps API [9] to convert an address to two points of GPS's coordinates [10]: latitude and longitude.

2) Software Engineering**:**

a. We will use the Spring Boot Framework for implementing the server side, that he is a reliable and efficient framework. To learn this framework, we took an online course. Furthermore, because we are experimenting with this technology for the first time, we can start with a prototype server and develop a server version for release that is based on the server prototype version. When we make sure to add features gradually, and make sure to run tests to check the integrity of the server in each addition of a new feature.
b. To make a reliable and fast interaction between the server side and the maintenance client, and with the Android-based application, we will use REST [6] architectural style that defines a set of constraints and properties based on HTTP [14] Web Services. Furthermore, we will use JavaBeans [13] which is a class that encapsulate many objects into a single object, and they are serializable. So, it is only left for REST [6] to transfer this serializable object into JSON file and send or receive the objects to/from the clients.
c. We will implement an authentication mechanism that consists a login screen in the client side, and a special token provided by the server to each user who successfully login to the client, and we will authenticate each request to the server from users using the token provided to them.
d. Our client will be a website that any user can access with a URL from anywhere. To achieve this, we develop the client side using Angular 2 technology, so it can be run on several servers around

the world, thus making it possible to quickly access the client website from all over the world. To achieve a user-friendly interface for the client side we design our website using HTML and CSS, and we use the Bootstrap design package for Angular 2.

e. We will use a framework that implements the JPA (Java Persistence API) [8] specification. The JPA is a Java application programming interface specification that describes the management of relational data in applications. Using this framework, we can perform CRUD [7] (i.e. create, read, update and delete) operations on the database's tables while keeping rational persistence and writing a minimum of code. Additionally, this framework will give us the power to change and expand tables in the database very easily because this framework is creating tables in the database automatically according to the classes and relationships between them that we have defined in the Java code.

1.6 Software Architecture

1.7 Overview

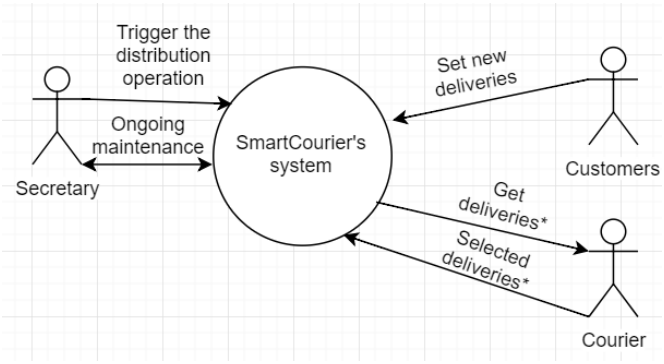

Fig.1: Overview

* A courier request for a deliveries list (such a list was created for each courier using the distribution algorithm), and then the courier selects the deliveries that he is willing to provide. Then the tool will assign these deliveries to the courier.

1.8 SmartCourier system internal view

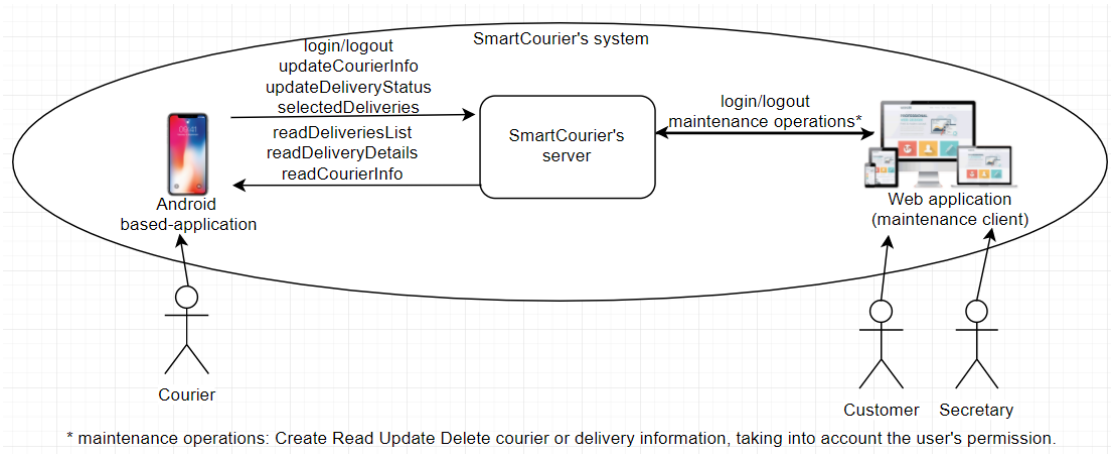

Fig.2: SmartCourier system internal view

1.9 SmartCourier server internal view

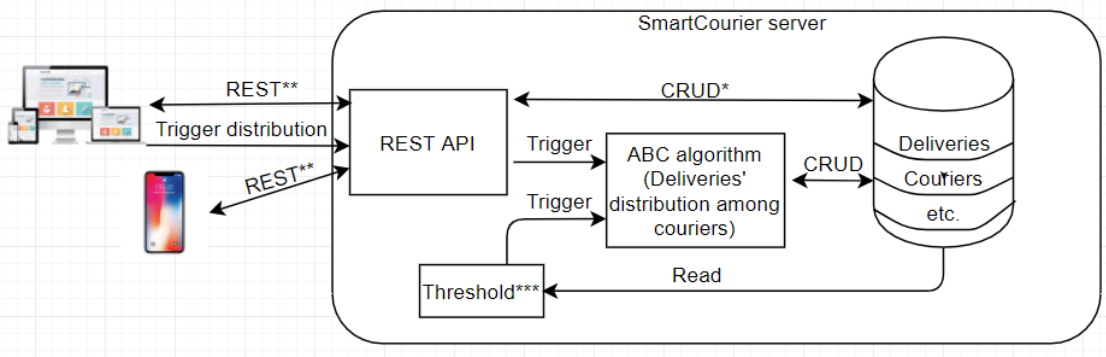

Fig.3: SmartCourier server internal view

* CRUD – Create Read Update Delete [7]: Performing basic operations on the tables of the database. These basic operations will be performed using a framework that is implementing the Java Persistence API **(**JPA**)** [8]**.**
** REST: For more information see 1.4 under 2. b.
*** Threshold: More details about this in section 3.5.

2. LITERATURE OF RELATED WORK

2.1 Introduction

- Term swarm in general refer to any restrained collection of interacting agents or individuals.
- Two fundamental concepts self-organization and division of labor are necessary and sufficient properties to obtain swarm intelligent behavior.

2.2 Behavior of Honey Bee Swarm

Three essential components of forage selection:

- Food Sources: The value of a food source depends on many factors such as its proximity to the nest, its richness or concentration of its energy, and the ease of extracting this energy.
- Employed Foragers: They are associated with a food source which they are currently exploiting or are "employed" at. They carry with them information about this food source, its distance and direction from the nest, the profitability of the source and share this information with a certain probability.
- Unemployed Foragers: They are continually at look out for a good source to exploit. There are two types of unemployed foragers: scouts, searching the environment surrounding the nest for new food source and onlookers waiting in the nest and establishing a food source through the information shared by employed foragers.
- The model defines two leading modes of the behavior
  - Recruitment to a nectar source.
  - The abandonment of a source.

2.3 Basic Self Organization Properties

- Positive feedback: As the nectar amount of food sources increases, the number of onlookers visiting them increases, too.
- Negative feedback: The exploitation process of poor food source is stopped by bees.
- Fluctuations: The scouts carry out a random search process for discovering new food sources.

- Multiple interactions: Onlookers watch the dances of employed bees and choose food sources depending on dances.

2.4 Artificial Bee Colony Algorithm (ABC Algorithm)

2.5 Introduction [1]

In computer science and operations research, the artificial bee colony algorithm (ABC) is an optimization algorithm based on the intelligent foraging behavior of honey bee swarm, proposed by Karaboga in 2005.

2.6 Algorithm [5]

In ABC, a population-based algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees is equal to the number of solutions in the population. At the first step, a randomly distributed initial population (food source positions) is generated. After initialization, the population is subjected to repeat the cycles of the search processes of the employed, onlooker, and scout bees, respectively. An employed bee produces a modification on the source position in her memory and discovers a new food source position. Provided that the nectar amount of the new one is higher than that of the previous source, the bee memorizes the new source position and forgets the old one. Otherwise she keeps the position of the one in her memory. After all employed bees complete the search process, they share the position information of the sources with the onlookers on the dance area in the hive. Each onlooker evaluates the nectar information taken from all employed bees and then chooses a food source depending on the nectar amounts of sources. As in the case of the employed bee, she produces a modification on the source position in her memory and checks its nectar amount. Providing that its nectar is higher than that of the previous one, the bee memorizes the new position and forgets the old one. The sources abandoned are determined and new sources are randomly produced to be replaced with the abandoned ones by artificial scouts.

2.7 Pseudo code [5]

- Initial food sources are produced for all employed bees.
- The best food source found so far is registered.
- REPEAT
  - Each employed bee goes to a food source in her memory and determines a closest source, then evaluates its nectar amount and dances in the hive
  - Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source. After choosing a neighbour around that, she evaluates its nectar amount.
  - Abandoned food sources are determined and are replaced with the new food sources discovered by scouts.
  - The best food source found so far is registered.
- UNTIL (requirements are met)

2.8 Clarification [2]

- Each cycle of search consists of three steps: moving the employed and onlooker bees onto the food sources and calculating their nectar amounts; and determining the scout bees and directing them onto possible food sources.
- A food source position represents a possible solution to the problem to be optimized.
- The amount of nectar of a food source corresponds to the quality of the solution.
- Onlookers are placed on the food sources by using a probability-based selection process.
- As the nectar amount of a food source increases, the probability value with which the food source is preferred by onlookers increases, too.

- The scouts are characterized by low search costs and a low average in food source quality.
- The selection is controlled by a control parameter called “limit”.
- If a solution representing a food source is not improved by a predetermined number of trails, then that food source is abandoned, and the employed bee is converted to a scout.

2.9 Control Parameters of ABC Algorithm [2]

- Swarm size
- Limit
- Number of onlookers: 50% of the swarm
- Number of employed bees: 50% of the swarm
- Number of scouts: 1

2.10 Exploration vs Exploitation [3]

- Onlookers and employed bees carry out the exploitation process in the search space.
- Scouts control the exploration process.

2.11 Basic Self Organization Properties [3]

- Positive feedback: As the nectar amount of food sources increases, the number of onlookers visiting them increases, too.
- Negative feedback: The exploitation process of poor food source is stopped by bees.
- Fluctuations: The scouts carry out a random search process for discovering new food sources.
- Multiple interactions: Bees share their information about food sources with their nest mates on the dance area.

2.12 Java source Code [5]

```
Static beeColony bee = new beeColony();
for(run=0;run<bee.runtime;run++)
{
        bee.initial();
        bee.MemorizeBestSource();
        for(iter=0;iter<bee.maxCyclel;iter++)
        {
                bee.SendEmployedBees();
                bee.SendOnlookersBees();
                bee.MemorizeBestSource();
                bee.SendScoutBees();
        }
}
```

3. APPROACH- IMPLEMENTING THE ABC ALGORITHM

To implement the ABC Algorithm in our project we need to adjust the basic concepts of the ABC algorithm to fit to our project.

3.1 Defining the Core Concepts of Our Algorithm:

- Division: A division is a subset of deliveries out of the total deliveries in the preferred region of the courier. That is, the number of divisions in a region is equal to the number of couriers in that region.
- Distribution: Distribution consist of many divisions, this corresponds to the food source in the ABC Algorithm. Each distribution, like a food source, has a value that is dependent on many factors.
- Factors: Each distribution will hold a list of factors that will be used later to evaluate the distribution by its employed bee and by the onlooker bee.
- Fitness: This is the fitness or the value of a distribution. Higher fitness means better probability to be chosen as a final solution. The algorithm will calculate the fitness of each distribution using his factors by using the following equations:

$$\forall\, i = 1, \dots, \#OfFactors.\ \overline{factors'Probabilities[index][i]} = \frac{distribution'sFactors[index][i]}{HighestPossibleFactor'sValue[i]}.$$

$$factors'Probabilities[index][i] = 1 - \overline{factors'Probabilities[index][i]}.$$

$$distributions'Fitneeses[index] = \sum_{i=1}^{numOfFactors} factors'Probabilities[index][i] \cdot i.$$

- When 'index' is the index of the distribution that its fitness is been calculated.
- $HPFV[i] = HighestPossibleFactorValue's[i]$ - The highest possible value of factor 'i'.
- $factors'Probabilies[i]$ Holds the normalized factor value, so it is in the range [0,1].
- Factor with higher index (i index) have more positive effect on the distribution's fitness.

3.2 Algorithm's Parameters

Without limiting the parameters' values, we will mark each parameter with a letter to make it easier to calculate the algorithm time complexity:

Deliveries – for each region we mark the maximum possible number of deliveries as **D**.

Couriers – We define **C** as the maximum number of couriers in a region. That means that the maximum number of divisions in region is also **C**, since each division belongs to one courier and each courier has only one division.

3.3 Defining the Factors for Evaluating a Distribution

For each distribution, we will calculate the fitness value using these two factors:

Distance – This factor will hold the average of all the averages of the driving distances between each of the deliveries' addresses in each division within the distribution. To calculate the complement of the factor's probability the algorithm will divide this average with the largest possible driving distance between two deliveries inside any division within the distribution, and the factor's probability will be 1 minus this result. In conclusion, the time complexity for calculating the probability of the Distance factor for each generated distribution is the number of edges in a complete graph with D nodes: $O(D^2)$.

Urgency – The algorithm should distribute the load of urgent deliveries between couriers. The reason for this is because the algorithm should prefer to distribute those urgent deliveries to several couriers to make faster deliveries in parallel, and not to depend on one courier to make those urgent deliveries. For this factor we give lower importance in our system than the importance we give to Distance and Time factors, because we should load on one courier a lot of urgent deliveries that are close to each other in terms of driving distance. Therefore, this factor will have a lower effect on the final calculation of the distribution's

fitness. To calculate this distribution's factor the algorithm will count the number of urgent deliveries for each division (courier), and the factor will be equal to the maximum number of urgent deliveries in one of the distribution's divisions. To calculate the complement probability of this factor the algorithm will divide the that maximum value with the total number of urgent deliveries in the distribution's region. In conclusion, the time complexity for calculating the probability of the Urgency factor for each generated distribution is the number of deliveries in the distribution's region: $O(D)$.

Hence the complexity of time of the Fitness of a distribution is: $O(D + D^2) = O(D^2)$.

Load – the load of the deliveries on each courier should be balanced as possible for the optimal distribution. To calculate this factor, the algorithm will add one to the 'delieveriesQuantity' counter that in each one of the distribution's divisions for each delivery that the algorithm inserts to that division. Then the algorithm will iterate each of the division 'delieveriesQuantity' counters and will find the maximum and minimum values of this counter. So, the Load factor will be equal to the maximum value minus the minimum value. Now, the algorithm will divide this subtraction with the maximum value that is possible for this subtraction. The maximum value of this subtraction is equal to the number of deliveries in the distribution's region. Now, the algorithm will calculate the factor's probability by performing 1 minus the ratio between subtraction and the maximum value of the subtraction, and this is necessary because lower values of that ratio means higher probability of load balancing of the deliveries among the courier (divisions). Time complexity for calculating the Load's probability: $O(C)$.
Hence the complexity of time of the Fitness of a distribution is: $O(D + D^2) = O(D^2)$.

3.4 Example for Calculating a Distribution's Factors and Fitness.

To clarify things, let us see an example for calculating the defined three factors. Given the following region and given five deliveries that are registered in this region. The addresses of that five deliveries are marked with purple numbers:

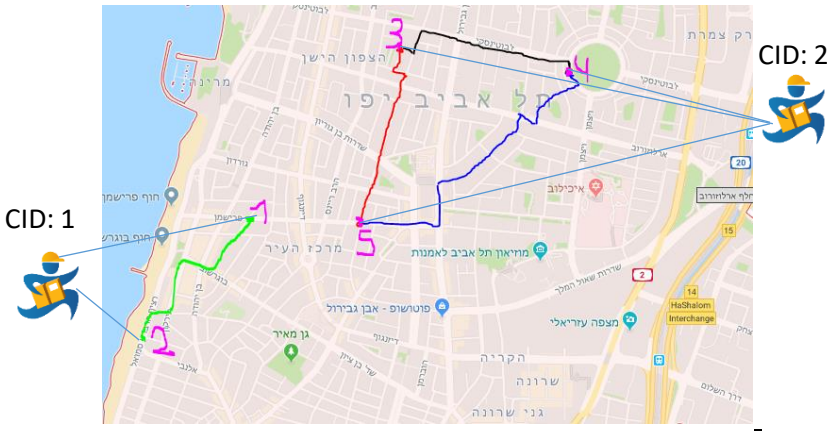

Fig. 4: deliveries registered in a region

We will define two couriers working in this region, so that means that distribution in this region has exactly 2 divisions (Division 1 include deliveries 3,4 and 5. Division 2 include deliveries 1 and 2). In the picture above, we can see a possible distribution of the five deliveries to the two couriers working in this region.

First the algorithm will calculate the probability for each of the three factors for this distribution:

1. **Urgency:** Let us define deliveries 3 and 5 as urgent. The Urgency factor will then be: $\text{Urgency}_{\text{factor}} = MaximumNumberOfUrgentDeliveriesInADivision = 2$

$$\overline{factors'Probabilities[index][1]} = \frac{distribution'sFactors[index][1]}{HighestPossibleFactor'sValue[1]} = \frac{2}{2} = 1$$

$$factors'Probabilities[index][1] = 1 - \overline{factors'Probabilities[index][1]} = 1 - 1 = 0$$

- When $distribution'sFactors[index][1] = \text{Urgency}_{\text{factor}} = 2$.
- When $HighestPossibleFactorValue's[1] = \#OfTotalUrgentDeliveries = 2$
- When 'index' is the index of the distribution that its fitness is been calculated.

The fitness of this factor is equal to 0 and is fine because in the example we have the worst possible load distribution of deliveries among couriers.

2. **Distance:** First, the algorithm will calculate the driving distance of each two deliveries in each of the divisions in the distribution:
   Division 1: $\text{DrivingDistance}(1,2) = 8.3\text{km}$

   Division 2: DrivingDistance(3,5) = 8.1km, DrivingDistance(3,4) = 6.9km, DrivingDistance(4,5) = 10.2km.

   Second, the algorithm will calculate the Distance factor using the driving distances:
   $\text{Distance}_{\text{factor}} = avg(avg(8.3) + avg(8.1 + 6.9 + 10.2)) = 8.35km$.

   Third, the algorithm will calculate the factor probability:

$$\overline{factors'Probabılıtıes[ındex][2]} = \frac{distribution'sFactors[index][2]}{HighestPossibleFactor'sValue[2]} = \frac{8.35km}{19km} = 0.4394736$$

$$factors'Probabilities[index][2] = 1 - \overline{factors'Probabılıtıes[ındex][2]} = 0.5605264.$$

   - When $distribution'sFactors[index][2] = \text{Distance}_{\text{factor}} = 8.35\text{km}$.
   - When $HighestPossibleFactorValue's[2] = DrivingDistance(2,4) = 19km$.
   - When 'index' is the index of the distribution that its fitness is been calculated.

3. **Load:**

$\text{delieveriesQuantity(Courier1=delieveriesQuantity(CID1)} = 2, \text{delieveriesQuantity(CID2)} = 3$

$\text{Load}_{\text{factor}} = \text{MAX(delieveriesQuantities)} - \text{MIN(delieveriesQuantities)} = 3 - 2 = 1.$

$$\overline{factors'Probabılıtıes[ındex][3]} = \frac{distribution'sFactors[index][3]}{HighestPossibleFactor'sValue[3]} = \frac{1}{5} = 0.2$$

$$factors'Probabilities[index][3] = 1 - \overline{factors'Probabılıtıes[ındex][3]} = 1 - 0.2 = 0.8.$$

   - When $distribution'sFactors[index][3] = \text{Load}_{\text{factor}} = 1$.
   - When $HighestPossibleFactorValue's[3] = NumberOfDeliveriesInRegion = 5$.
   - When 'index' is the index of the distribution that its fitness is been calculated.

Finally, the fitness of this distribution will be: $distributions'Fitneeses[index] = \sum_{i=1}^{numOfParams} factorsProbabilies[i] \cdot i = \sum_{i=1}^{3} factorsProbabilies[i] \cdot i = 0 \cdot 1 + 0.5605264 \cdot 2 + \cdot 3 = 1.1210528$.

We got $distributions'Fitneeses[index] = 1.1210528$, and the maximum possible value for the fitness is $1 + 2 = 3$. So, this distribution will not be chosen as the final solution if there is a distribution with higher fitness.

3.5 Implementing the ABC Algorithm's Main Methods

As was explain in section 3.3, the time complexity for calculating the Fitness for a distribution is: $O(D^2)$. For each region the server will run the following main methods of our ABC algorithm:

bee.initial(): Initial food sources are produced for all employed bees: Number of distributions are generated randomly in each region (Associating each delivery with a random courier in that region). Time complexity: $O(D + C) \cdot NumOfdistributions) =^{*} O(D)$.

bee.MemorizeBestSource(): This method finds and stores the distribution with the maximum fitness. Time complexity: $O(NumOfdistributions) =* O(1)$.

bee.SendEmployedBees(): Each employed bee goes to a food source in her memory and determines a closest source, then evaluates its nectar amount and dances in the hive: The fitness of each distribution is being calculated. Time complexity: $O(D^2 \cdot NumOfdistributions) =^* O(D^2)$.

bee.SendOnlookersBees(): Each onlooker watches the dance of employed bees and chooses one of their sources depending on the dances, and then goes to that source. After choosing a neighbour around that, she evaluates its nectar amount: The onlooker bee produces a modification on a random distribution. The modification will be on two random divisions from that distribution. So that the modification will be a substitution between a random number of deliveries between those divisions. Then the onlooker bee will calculate the fitness of the modified distribution. If the fitness of the modified distribution is higher than the unmodified distribution, then the unmodified distribution will be replaced with the modified distribution. Otherwise do not do anything. This operation ensures that for a very high number of iterations performed in this algorithm, the algorithm will review the entire range of possible solution in this optimization problem. Time complexity: $O(D^2)$.

bee.SendScoutBees(): Abandoned food sources are determined and are replaced with the new food sources discovered by scouts: Each distribution has a trail counter that is initialized to zero when the distribution is generated for first time. Each time the algorithm generates a new modified distribution in the SendOnlookersBees method the algorithm compares the fitness of the new modified distribution with the fitness of the old unmodified distribution. If the fitness of the new modified distribution is not higher than the fitness of the old unmodified distribution the algorithm will increase the trail counter of the unmodified distribution by one. Otherwise, it will not increase any trail counter. The trail counter of the unmodified distribution indicates how much failed attempts have been made to improve the fitness of the unmodified distribution. In the SendScoutBees, the algorithm iterates for each trail of the distributions and finds out which trail's distribution has acceded the maximum trail. The algorithm will discard this distribution and initialize a new random distribution. This operation ensures that the algorithm will not fall into a local maximum fitness's value. Because in a local maximum, it will be hard to improve the fitness value. It is important to note that if the algorithm has discarded a local maximum, which happen to be the global maximum of the fitness's value, the algorithm is already stored this distribution in the MemorizeBestSource method. Additionally, this process ensures that on each iteration of the main loop the algorithm will discard of exactly one distribution. Time complexity: O(numberOfDistributions) = O(1).

Algorithm's total time complexity: Those main methods are performed inside a for loop that iterates numberOfDistributions times, so:

$$O\big((D^2 + C) \cdot \text{numberOfDistributions}\big) =^* O((D^2)).$$

* Assuming the number of distribution is a very small fixed number: $\text{numberOfDistributions} = O(1)$, and assuming: $C = o(D)$.

3.6 Triggers of the Algorithm

First, we distinguish between the three types of deliveries that exist in our database:

- Type 0: Deliveries that have not yet been assigned to a courier because they have not yet been distributed by the algorithm.
- Type 1: Deliveries were distribute by the algorithm to a courier but the courier has not courier has not yet confirmed that he is willing to distribute them.
- Type 2: deliveries were distributed by the algorithm to the courier and were confirmed by the courier that he is willing to distribute them.

- Type 3: Deliveries that have already been delivered to the destination and therefore will no longer play a role in the algorithm.

It is important to distinguish between those four types of deliveries because once a courier approves to make a delivery, the algorithm will not distribute this delivery any more.

Second, we will define a threshold value for each region that will define how much of deliveries from type 0 and type 1 we can have in the database for that region until we will trigger the algorithm to distribute the deliveries to couriers in that region. In other words:

- If Threshold > #OfTypes0And1Deliveries then distribute types 0 and 1 deliveries to couriers.
- Otherwise do nothing

Note: We will not set the threshold value to be too high to avoid the situation where some deliveries have not been assigned to a courier for a long period of time.
The algorithm will run continuously, meaning that each time the number of deliveries of type 0 plus type 1 in a specific region passes the threshold value we will distribute the type 0 and type 1 deliveries using the ABC algorithm to the couriers in that region. When the trigger is activated the algorithm will distribute the deliveries as follows: If there are deliveries of type 2 (assigned to a courier but did not reach their destination yet). The ABC algorithm will distribute the deliveries of type 0 and type 1 between the couriers while keeping the deliveries of type 2. Then, the algorithm will calculate the fitness of the distribution while taking into consideration the delivery of type 0, type 1 and type 2.

4. METHODOLOGY & PLANNING

4.1 Requirements Engineering

Throughout the project analysis phase, we made sure to stick to the process of requirements engineering:

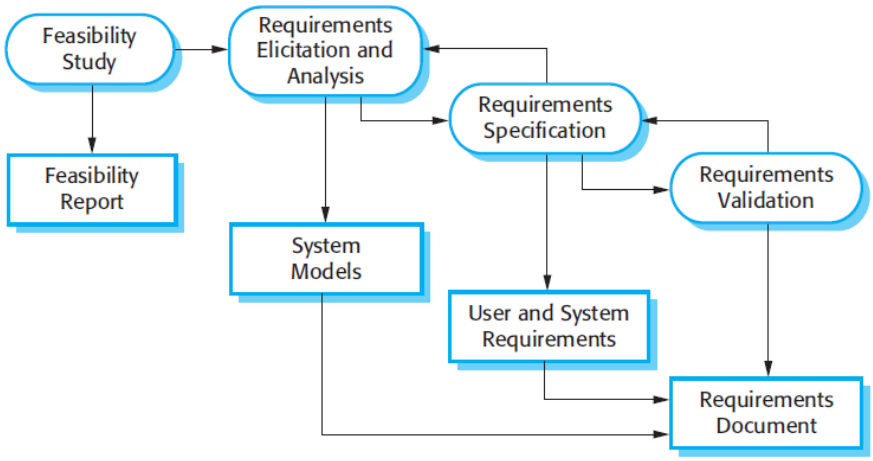

Fig. 5: requirements engineering

4.2 Activities

1. Requirements analysis & Project planning.
2. Designing (UML).
3. Developing the server-side (desktop application):

3.1. Reviewing and analyzing the server-side algorithms.
3.2. Definition of the algorithm(s) to solve the problems.
3.3. Coding and QA (QA will be performed using soap UI [11]).

4. Developing the client-side(web application)

4.1. Reviewing and analyzing the client-side algorithms.
4.2. Reviewing and analyzing the client-side algorithms.

4.3. Definition of the algorithm(s) to solve the problems.
4.4. Coding and QA.
5. Implementing server-client communication
6. Running of the project in real life examples.
7. Analysis of the results and conclusions.
8. Finite tuning of the algorithm if needed.
9. Testing the project in a real life while considering edge cases.

4.3 Scheduling

We both work part-time in high-tech. Therefore, it was very important to plan the right times to meet goals and show good results. To set schedules and meet them, we opened a calendar together and marked the days and hours of each week. In addition, the progress process is a significant part of the project, and it is important to show progress every week, so we set goals for each week of the semester and consistently met those goals.

4.4 Success factors

1. There will be no more than a threshold number of packages waiting to be distributed (More details in section (3.5)).
2. The algorithm must be time complexity efficient and must always bring a result that is among the 3% best results.
3. All deliveries will be handled eventually in a reasonable time.
4. Urgent deliveries must be at the top priority.

4.5 Conclusions

Our final applications need to be flexible, effective, and easy to use.

Therefore, in our project, we will use fast algorithms. We will use Monte Carlo algorithm [12] which produces a very good result in low-time complexity. For more details about making the applications flexible and easy to use please read section 1.4. Furthermore, the implemented ABC algorithm will save a lot of money that was supposed to be paid to the secretaries who would distribute the letters manually and of course would not come to efficient and accurate distributions as the algorithm will come.

5. DESIGN

5.1 Infrastructure and Design:

1. For developing the server, we will use Eclipse IDE with JDK.
2. For debugging the web-application, we will use Chrome DevTools [16].
3. For coding the web-application, we will use Editor ATOM [15].

5.2 Application Views:

5.2.1 Login screen:

Description: Login into the application.
Input: user name and password
Output: Depends on the correctness of the input.

1. The user name exists at the DB, and the password matches it:
   The user redirected to the main screen
2. The user name exists at the DB, and the password does not match it:
   An error bar with the proper message will pop out, the user will have the option to press “OK” and to try again.
3. The user name does not exist at the DB:
   An error bar with the proper message will pop out, the user will have the option to press “OK” and to try again.

5.2.2 Courier main screen:

Initialize screen: A list of couriers will be loaded from the server and will show up in a table.
Description: The Courier Main screen of the application where the user can watch the list of the couriers' details, choose what to do next:

1. Log out.
2. Create new courier.
3. Edit courier details.
4. Delete courier.
5. Watch a specific courier's deliveries.
6. Filter couriers' salaries according to year and month.
7. Go to All Deliveries screen

Input: The user chooses the desired option.
Output:

- For the option 1 the user will transfer to login screen.
- For the option 2 the user will transfer to Create New Courier screen.
- For the option 3 the user will transfer to Edit Courier screen.
- For option 4 a question bar will pop out, to make sure the user is certain he wants to delete the courier.
- For the option 5 the user will transfer to Courier's Deliveries screen.

5.2.3 Create new courier screen:

Initialize screen: A empty form with all the courier's fields will appear.
Description: add new courier.
Input: first and last name, user name, password, email address, phone number, etc.
Output: The user redirected to the main screen, and a successful message appears.

5.3 Edit Courier Screen:

Initialize screen: A pre-filed form with the courier's details will appear.
Description: Edit the courier's details.
Input: The details that are required to be changed.
Output: The user redirected to the main screen, and a successful message appears.

5.3.1 Courier's deliveries screen

Initialize screen: A list of deliveries for the chosen courier will be loaded from the server and will show up in a table.
Description: The Courier Deliveries screen of the application where the user can watch the list of the couriers' deliveries' details, and choose what to do next:
1- Log out.
2- Edit delivery details. .
3- Delete delivery.
Input: The user chooses the desired option.
Output:

- For the option 1 the user will transfer to login screen.
- For the option 2 the user will transfer to Edit Delivery screen.
- For option 3 a question bar will pop out, to make sure the user is certain he wants to delete the courier.

5.3.2 All deliveries screen

Initialize screen: A list of all the deliveries will be loaded from the server and will show up in a table.
Description: The Deliveries screen of the application where the user can watch the list of the all the deliveries details, and choose what to do next:
1- Log out.
2- Create new delivery.
3- Delete delivery.
4- Edit delivery details.

5- Distribute Deliveries
Input: The user chooses the desired option.
Output:

- For the option 1 the user will be transferring to login screen.
- For the option 2 the user will be transferring to Create New Delivery screen.
- For option 3 a question bar will pop out, to make sure the user is certain he wants to delete the delivery.
- For the option 4 the user will be transferring to Edit Delivery screen.
- For the option 5 the screen will be updated with the assignation of each delivery to a courier.

5.3.3 Create new delivery screen:

Initialize screen: An empty form with all the Delivery's fields will appear.
Description: add new delivery.
Input: All the delivery fields (There will be a combo box with all the courier's ID in the chosen region. The delivery will assign to the chosen courier's ID).
Output: The user redirected to the All Delivery screen. And a successful message appears.

5.4 Edit delivery screen:

Initialize screen: A pre-filed form with the delivery's details will appear.
Description: Edit the delivery's details.
Input: The details required to be changed (There will be a combo box with all the courier's ID in the chosen region. The delivery will assign to the chosen courier's ID).
Output: The user redirected to the All Delivery screen. And a successful message appears.

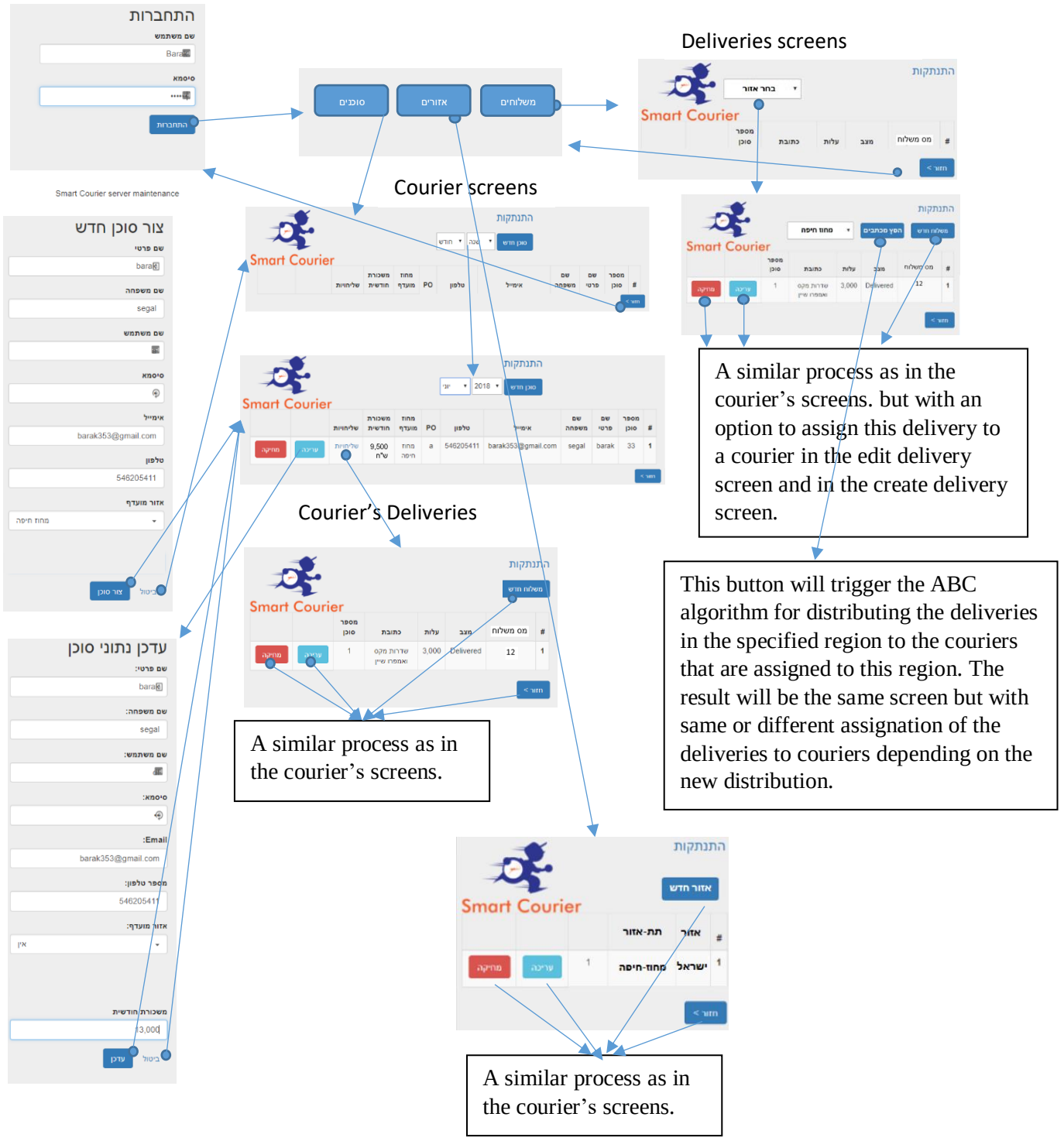

Fig.6: Screen diagram

5.5 UML

5.5.1 Use Case

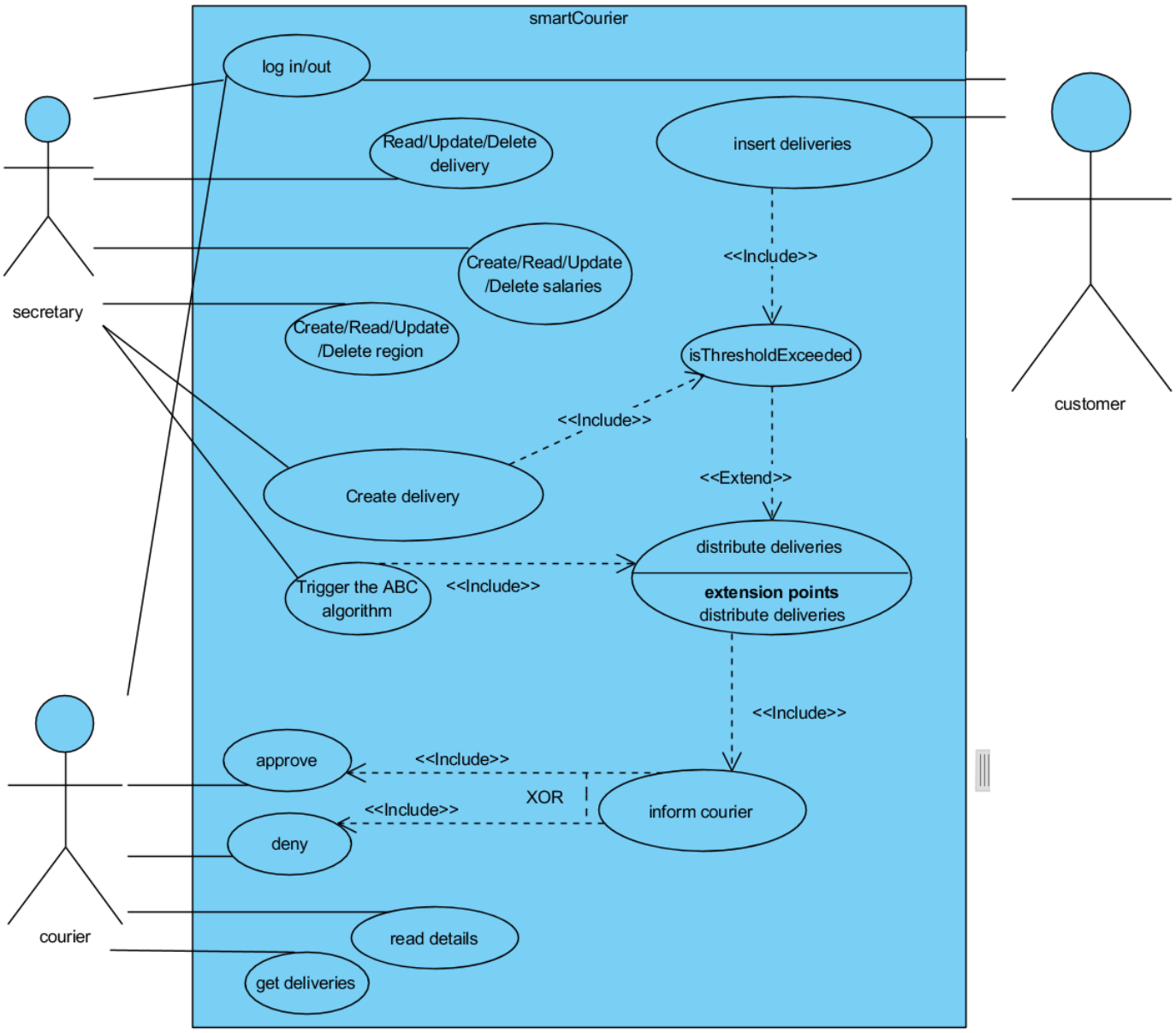

Fig.7: Use case

5.6 Class Diagram

5.6.1 Maintenance Client

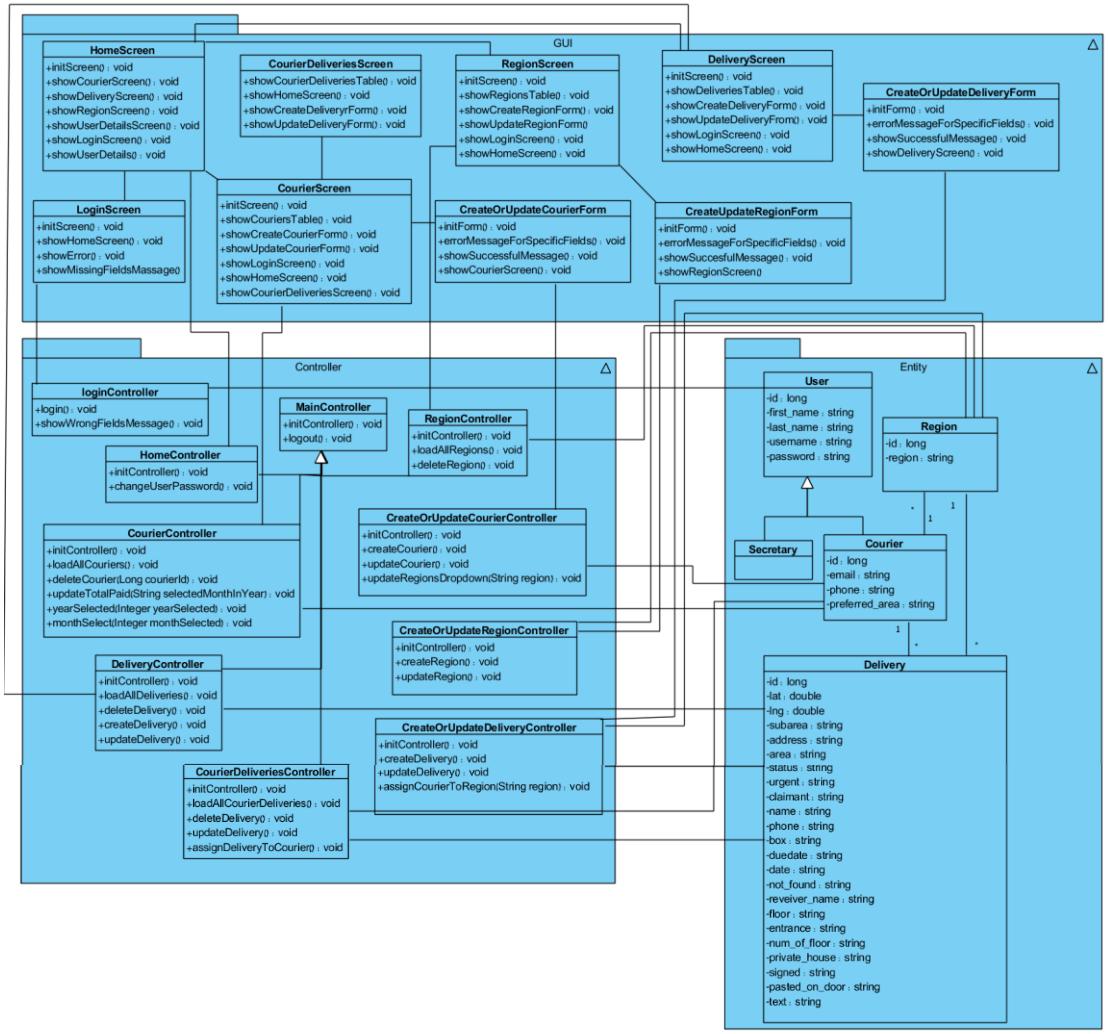

Fig. 8: maintenance client

5.6.2 Server

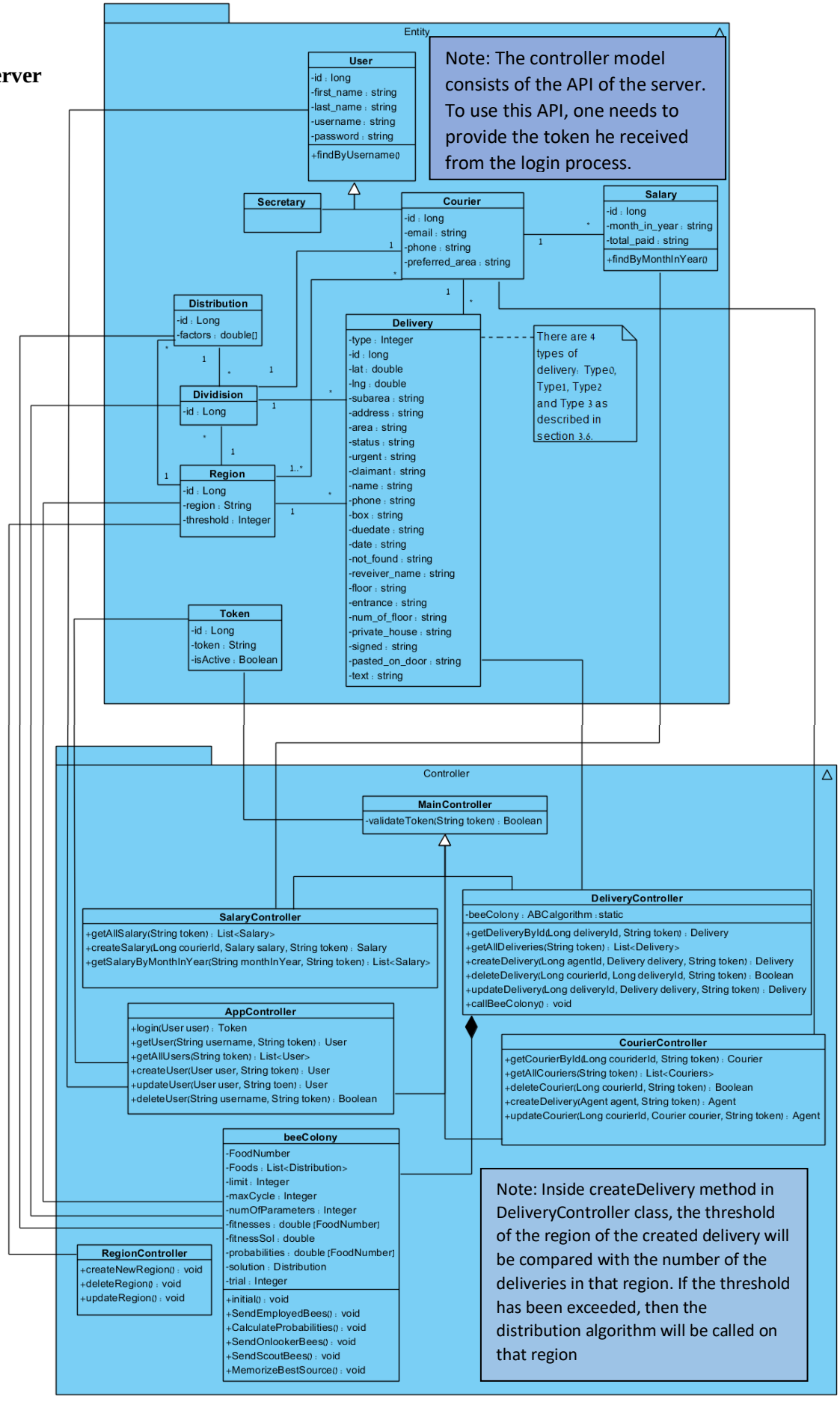

Fig.9: Server

5.7 Testing Plan

#Test	Test type	Test name	Description	Expected results
1	login	Login_FailByID	User login with wrong user name	Error message: “Wrong username, please try again.”
2	login	Login_FailByID	User login with wrong password	Error message: “Wrong password, please try again. After three times failing your username will be temporarily blocked.”
3	login	Login_Fail3Times	User login with wrong password 3 times	Error message: “Wrong password. Your password has been blocked. Try again in 15 minutes.”
4	login	Login_Fail4Times	User login with wrong password for the fourth time	Error message: “Your password has been blocked. Try again in 15 minutes.”
5	login	Login_succ	User login with correct user name and password	Go to Main screen
6	create courier	CreateCourier_succ	User insert all the data correctly and press Add Courier	Confirmation message: “The courier has been added successfully.” Press OK and user will be back at the couriers screen
7	create courier	CreateCourier_fail	User insert incorrect data	Show red messages next to each wrong box.
8	Update courier	UpdateCourier_succ	User replace previous data with new correct data	Confirmation message: “The courier has been updated successfully.” Press OK and user will be back at the couriers screen
9	Update courier	UpdateCourier_fail	User replace previous data with new incorrect data	Show red messages next to each wrong box.
10	Create delivery	CreateDelivery_succ	User insert all the data correctly and press Add delivery	Confirmation message: “The delivery has been added successfully.” Press OK and user will be back at the deliveries screen
11	Create delivery	CreateDelivery_fail	User insert incorrect data	Show red messages next to each wrong box. Press OK and user will be back at the deliveries screen
12	Update delivery	UpdateDelivery_succ	User replace previous data with new correct data of type 0 or 1 delivery	Confirmation message: “The delivery has been update successfully.” Press OK and user will be back at the deliveries screen

13	Update delivery	UpdateDelivery_failType	User replace previous data with new correct data of type 2 or 3 delivery	Error message: “The delivery has been sent! There is no option to edit it!” Press OK and user will be back at the delivery screen
14	Update delivery	UpdateDelivery_failData	User replace previous data with new incorrect data of type 0 or 1 delivery	Show red messages next to each wrong box.
15	Distribute Deliveries	DistributeDeliveries_Succ	User click on Distribute Deliveries in All Deliveries screen.	
16	Distribute Deliveries	DistributeDeliveries_Fail	User click on Distribute Deliveries in All Deliveries screen.	
17	logout	logout	User click on logout button in any screen.	User will be logged out from the client web-application. The user will not have access to server API any more. In the data base the status of the user will be “off”
18	Salaries by date	SalariesByDate_succ	User choose month and year in Courier Main screen	The salaries for the chosen month and year will appear in the table for all the couriers who got paid in that month and year.
19	Salaries by date	SalariesByDate_succ	User choose month and year in Courier Main screen	All deliveries screen will show up.

Table 1: Testing plan

6. REFERENCES

[1] *Karaboga, Dervis (2005). "An Idea Based on Honey Bee Swarm For Numerical Optimization" (PDF).*

[2] E. Bonabeau, M. Dorigo, G. Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems", New York, NY: Oxford University Press, 1999.

[3] D. Karaboga , B. Basturk "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J Glob Optim (2007) 39:459–471

[4] D.Karaboga "An idea based on honey bee swarm for numerical optimization" TR-06, October 2005

[5] *Artificial Bee Colony (ABC) Algorithm Homepage, Turkey: Intelligent Systems Research Group, Department of Computer Engineering, Erciyes University*

[6] REST. Representational State Transfer : https://en.wikipedia.org/wiki/Representational_state_transfer

[7] CRUD. Create, read, update and delete: https://en.wikipedia.org/wiki/Create,_read,_update_and_delete

[8]Java Persistence API (JPA): https://en.wikipedia.org/wiki/Java_Persistence_API

[9] Google Maps API: https://en.wikipedia.org/wiki/Google_Maps and https://developers.google.com/maps/documentation/

[10] Geographic coordinate system https://en.wikipedia.org/wiki/Geographic_coordinate_system

[11] SoapUI: The world's Most Popular API Testing Tool: https://en.wikipedia.org/wiki/SoapUI

[12] Monte Carlo algorithm: https://en.wikipedia.org/wiki/Monte_Carlo_algorithm

[13] JavaBeans: https://en.wikipedia.org/wiki/JavaBeans

[14]Hypertext Transfer Protocol (HTTP): https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

[15] Chrome DevTools: https://developers.google.com/web/tools/chrome-devtools/

[16] Atom editor: https://atom.io/