**REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE**

**MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE**


**UNIVERSITE MOHAMED BOUDIAF - M'SILA**
**FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE**

**DEPARTEMENT D'INFORMATIQUE**


**MEMOIRE de fin d'étude**

**Présenté pour l'obtention du diplôme de MASTER**

**Domaine** : Mathématiques et Informatique

**Filière** : Informatique

**Spécialité** : Système d'Information et Génie Logiciel (SIGL)


**Par** : Bouchelaleg Fouzia

Chiba Abdechafiee


## SUJET

# Conception Et Réalisation d'un API java pour Gestion des Emplois du Temps

**Soutenu publiquement le : .. / .. /2021 devant le jury composé de** :

| | | |
|---|---|---|
| Dr. Debbi Hicham | Université de M'sila | Président |
| Dr. Heraguemi KamelEddine | Université de M'sila | Rapporteur |
| Dr. Lakhal Meftah | Université de M'sila | Examinateur |

**Promotion: 2020 /2021**

**PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA**

**MINISTRY OF HIGHER EDUCATION AND**

**SCIENTIFIC RESEARCH**


**UNIVERSITE DE MOHAMED BOUDIAF- M'sila**

**FACULTY OF MATHEMATICS**

**AND COMPUTE SCIENCE**

**COMPUTER SCIENCE DEPARTMENT**


## A Dissertation in Fulfillment

## For the Requirements of the Degree of Master

**DOMAIN**: Mathematics and Computer Science

**BRANCH**: Computer Science

Information System and Software Engineering (SIGL)


**Through**: Bouchelaleg Fouzia

Chiba Abdechafiee

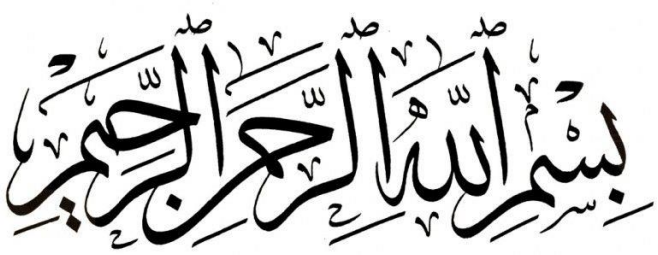

## SUBJECT

# Design And Realization Of A Java API For Schedule Management

**Publicly supported on: .. / .. /2021 in front of the jury composed of:**

| | | |
|---|---|---|
| Dr. Debbi Hicham | M'sila University | President |
| Dr. Heraguemi KamelEddine | M'sila University | Reporter |
| Dr. Lakhal Meftah | M'sila University | Examiner |

**2020 /2021**




# *Dedications*

I dedicate this work:

To my dear father and my dear mother.

To my dear sisters Nour elhouda and Nadjoua.

To my sister's husband and her son.

To all my family and my friends.

To all who have sacrificed their time for science and to all who use science for the good and prosperity of mankind

*Fouzia*

I dedicate this work:

To my dear father and my dear mother.

To my dear brothers and sisters.

To all my family and my friends.

*Abdechafié*

# *ACKNOWLEDGEMENT*

In the name of God, the Most Gracious, the Most Merciful, and praise be to God, Lord of the Worlds, who helped us to finish this research and to come out with it in this integrated way. It contains the ambition of young people who dream that their Arab nation is like a mole among nations.

I thank God for his bounty, as he allowed us to accomplish this work thanks to him. Praise be to God first and last and then to our generous parents for all their efforts since our birth until these beautiful moments. You are everything. We love you in God the most.

We also thank the honorable professor who supervises our work\ Dr. Heraguemi Kamal Eddine, who spared no effort to help us and guide us. He was like a second father to us. He opened his office to us with open arms, as is his custom with all students of knowledge, and we used to sit with him for long hours, guiding us. He did not find any embarrassment in that, and he used to urge us to search, desire us to do it, and strengthen our resolve for it. He has a reward from God, and we have all the appreciation, may God protect him, and enjoy health and wellness, and benefit from his knowledge.

All thanks to teachers for their acceptance to evaluated our sample work

We also thank those in charge of the University of Mohamed Boudiaf M'sila, headed by His Excellency Dr. / Bedari kamal the President of the University, and His Excellency the Dean of the College, Dr. /Bourahla mustapha , and may God grant them all the best for their interest in the students of the Faculty of Mathematics and Media The highest in general and the automated media department in particular

# *Table of Contents*

## *List of Figures*

# GENERAL INTRODUCTION

# GENERAL INTRODUCTION

At the height of the scientific and technological development in the field of automated media, software and digitization, which is witnessing great progress at the present time and because of its positive impact on our daily life for the individual, which contributed to facilitating the various tasks assigned to the individual in many practical aspects, It can provide more accurate and faster results compared to the average person in many cases and it is the pillar of the global economy and the development of the work of various institutions.

Technology has been closely linked to human development and social change in its various manifestations, regardless of its connection, and by means of variable and varied information in its scientific and practical specificity, technology has emerged as an important factor in social change, and thus in public administration services for various institutions, and a person can employ his techniques and manufactures in changing his conditions. And in adapting his conditions according to his predictions and requirements. [1] , And its exploitation by the administration and public institutions is an inevitable fact for the purpose of exploiting it in professional life management programs.

The schedule creation problem is a typical scheduling problem that seems like a daunting job in every academic institute once or twice a year. In the previous days, scheduling was done manually with one person or group participating in the scheduling task manually, which required a lot of effort and time. Planning timelines is one of the most complex and error-prone applications. Scheduling is the task of creating a schedule while meeting some limitations.

There are two basic types of restrictions, soft restrictions and hard restrictions. Soft restrictions are those that if we violate them in scheduling the output, they remain valid, but the hard ones are those that if we violate them; The table is no longer valid. The search area for a table problem is very vast, there are many solutions in the search space and some of them are not possible. Feasible solutions here mean those that do not violate strict restrictions and also try to satisfy soft limitations. We need to choose the most suitable solution from the possible solutions.

Most favorable here means those who do not violate the soft restrictions to a large extent. In this project, work was done to finalize a program to set a schedule for educational institutions using genetic algorithms, by paying attention to strict restrictions and ensuring that soft restrictions are followed as closely as possible.

This thesis is composed in four chapters as follow:

In chapter on we present Different definitions about time table, Approaches to Automated Timetabling the second chapter presents the Explanation of the genetic algorithm The third one talks about API and the last one gives a brief presentation of our proposed system.

# CHAPTER 1 :TIME TABLE

# 1. Introduction

Sometimes, the words schedule, sequence and timetable are loosely used as if were synonymous. But, there can be certain distinctions between these terms observed in the literature [ [2] [3], [4], [5], [6] [7]].

A *timetable* shows when particular events are to take place. It does not necessarily imply an allocation of resources. Thus, a published bus or train timetable shows when journeys are to be made on a particular route or routes. It does not tell us which vehicles or drivers are to be assigned to particular journeys. The allocation of vehicles and drivers is part of the scheduling process. Although timetabling is strictly the design of the pattern of journeys, this pattern may be devised as part of a process which bears in mind whether it is likely that an efficient schedule may be fitted to the resulting journey pattern.

In the rail domain, the term timetabling is often used to refer the construction of a path (with times) for a train through a system. A class timetable shows when particular events are to take place. In an infants' school where a single teacher is responsible for all the activities of a particular class, and where these activities all take place in the same room, a timetable is nothing more than a statement as to the times at which particular activities will take place. By contrast, a university examination timetable will normally include room assignments drawn up in the knowledge of group sizes and of special facilities needed. A university class timetable has also to take into account the availability of individual lecturers. The activities of drawing up examination and university class timetables may be considered as scheduling activities.

A *sequence* is simply an order in which activities are carried out. For example, the order in which jobs are processed through the machines of a factory, if jobs pass through each machine in the same order, is a sequence. Sequencing may take into account costs related to one particular job being followed by another (e.g., machine conversion costs). The problem of sequencing jobs in these circumstances is known as a flow shop problem.

A *schedule* will normally include all the special and temporal information necessary for a process to be carried out. This will include times at which activities are to take place, statements as to which resources will be assigned where, and work plans for individual personnel or machines.

The goal of scheduling in its broadest sense is to solve practical problems relating to the allocation, subject to constraints, of resources to objects being placed in space-time, using or developing whatever tools may be appropriate. The problems will often relate to the satisfaction of certain objectives. A.Wren defines scheduling, timetabling, sequencing and rostering [3] as follows

# 2. Definitions

## 2.1. Scheduling

Scheduling is the allocation, subject to constraints, of resources to objects being placed in space-time, in such a way as to minimize the total cost of some set of the resources used. [8]

As an example, we take the scheduling of transport or delivery that depends on reaching the goal at the lowest possible cost, and accordingly the means of transportation is directed to reduce the number of vehicles or drivers (human and material resources).

**Example 2— Timetable Scheduling** [9]

**Weekly schedule**

**Name:**

| Time / period | Monday | Tuesday | Wednesday | Thursday | Friday | Saturday |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

*Figure 1 Time table*

## 2.2. Timetabling

Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives [8]

The timing of the exams is the best example of this, as it adheres to the conditions of space-time and works to meet the goal of giving the best scheduling in conjunction with the requirements of the student and the professor and the availability of the necessary equipment

## 2.3. Sequencing

Sequencing is the construction, subject to constraints, of an order in which activities are to be carried out or objects are to be placed in some representation of a solution [8]

Examples of sequencing are flow-shop scheduling and the travelling salesman problem.

## 2.4. Rostering

Rostering is the placing, subject to constraints, of resources into slots in a pattern. One may seek to minimize some objective, or simply to obtain a feasible allocation. Often the resources will rotate through a roster [8]

## 3. Different problems and formulations

A large number of variants of the timetabling problem have been proposed in the literature, which differ from each other based on the type of institution involved (university or school) and the type of constraints. We classify the timetabling problems into three main classes:

**School timetabling**: The weekly scheduling for all the classes of a school, avoiding teachers meeting two classes at the same time, and vice versa.

**Course timetabling:** The weekly scheduling for all the lectures of a set of university courses, minimizing the overlaps of lectures of courses having common students.

**Examination timetabling:** The scheduling for the exams of a set of university courses, avoiding overlap of exams of courses having common students, and spreading the exams for

The students as much as possible

The school timetable describes several basic and non-essential conditions that must be required for each subject a number of hours that must be respected according to the subject's parameter and importance, often the number of hours for each subject taught per week is assigned nationally, and each specific class has a room in which it is held There is a professor for each semester. Often professors are determined in advance on the appointment process Each class or teacher must not participate in more than one meeting time. Often times, it is required that each teacher have at least a morning or noon. There may be many other similar restrictions.

The difference between the schedule of the (university) and the school (high school) course is that school teachers always have more than one semester, while in universities, a professor may teach only one course. Moreover, university courses can contain regular students, while school classes are separate groups of students. If two of the classrooms have students in common then they conflict, and they cannot or should not be Scheduled for the same period. In addition, a large hall should be available in order to accommodate students, and display equipment and other necessary tools should be provided

The problem of preparing a schedule for the exam involves teaching a specific subject. The number of exams is usually one for each subject at the end and during a certain period of time.

Examination schedule problems can be formulated with mentioning some acceptable contradictions that would define different conditions and important characteristics. We can mention them:

- There is one test for each subject
- There are several types of restrictions, such as passing one subject every day. It is also possible to pass more than one test in a room
- Avoid preparing tests for two materials with large labs in one day
- Choose the appropriate time and avoid stress during breakfast and lunch.

There are also several restrictions that depending on the university and students are also included in the conditions for completing a good exam schedule.

## 4. University course timetabling

The problem of scheduling the undergraduate course is subject to many constraints that are usually divided into two categories, hard and soft. In scheduling a group of lectures for each session within a certain number of rooms and time periods. If there are students involved in two courses, they conflict, and they cannot be scheduled in the same period. Moreover, school teachers always teach more than one class, while in universities, a professor usually teaches only one course. In addition, in the university problem, the availability (and size) of rooms plays an important role, while in the school problem it is often neglected because, in most cases, we can assume that each class has its own room.

As for the soft restrictions, which are those that are desirable but not completely necessary, for example, the allocation and lack of time between events that may embarrass the student at times and take more effort from him. On separate days in order to be well absorbed by the student, which is based on the principle of coordination in the events and their succession according to specific conditions. The teacher also requires the availability of a room and the necessary equipment that in turn facilitates and completes his task in teaching.

The administration works in coordination with the head of each department who represents its students on the intersection of soft and hard restrictions that depend on the available human and material resources that contribute better to preparing the schedule for the inclusive session, but this remains with it the problem of searching for what we have touched upon, we begin to describe the basic formulation of the problem. Next, we introduce the optimization problem and discuss the variables of the problem. Finally, we present the solution techniques and methods

### 4.1. Soft-constraints (flexible):

- More or less equal load is given to all faculties
- Required time (hours per week) is given to every Batch
- Room occupancy: Two distinct lectures cannot take place in the same
- room in the same period.
- Lectures of courses in the same curriculum must be all
- scheduled at different times.

### 4.2. Hard-constraints (rigid):

- There should not be any single instance of a faculty taking two classes simultaneously
- A class group must not have more than one lectures at the same time
- Room capacity (soft): The number of students that attend a course must

- be less than or equal to the number of seats of all the rooms that host its lectures.

### 4.3. Basic Search Problem

There are various definitions of the course timetabling problems. In [2] [5] course timetabling is defined as the following search problem:

**Definition:** *(course timetabling).* There are $q$ courses $K_1,K_2,\ldots K_q$, and for each $i$, course $K_i$ consists of $k_i$ lectures. There are $r$ *curricula* $S_1,S_2,\ldots S_r$, which are groups of courses that have common students. This means that courses in $S_l$ must be scheduled all at different times. The number of periods is $p$, and $l_k$ is the maximum number of lectures that can be scheduled at period $k$ (i.e., the number of rooms available at period $k$). The formulation is the following:

find $y_{ik}$ $(\forall\ i = 1,\ldots q;\ \forall\ k = 1,\ldots p)$ , so that

- $\forall\ i = 1,\ldots q \sum\{ y_{ik} \mid k = 1,\ldots p\} = k_i$
- $\forall\ k = 1,\ldots p \sum\{ y_{ik} \mid i = 1,\ldots q\} \leq l_k$
- $\forall\ k = 1,\ldots p\ \forall\ l = 1,\ldots r \sum\{ y_{ik} \mid i \in S_l\} \leq 1$
- $\forall\ i = 1,\ldots q\ \forall\ k = 1,\ldots p\ y_{ik} \in \{0,1\}$

where $y_{ik} = 1$ if a lecture of course $K_i$ is scheduled at period $k$, and $y_{ik} = 0$ otherwise.

The first constraint imposes that each course is composed of the correct number of lectures. The second constraint enforces that at each time there are notmore lectures than rooms. The third constraint prevents conflicting lectures to be scheduled at the same period. Problem from Definition
can be shown to be NP-complete through a simple reduction from the graph coloring problem (see [2]). A formulation equivalent to Definition is based on the *conflict matrix* instead of on the curricula. The conflict matrix $C_{q\times q}$ is a binary matrix such that $c_{ij} = 1$ if courses $K_i$ and $K_j$ have common students, and $c_{ij} = 0$ otherwise. In ( [2], [5]), the course timetabling problem also includes the following objective function:

**Definition :** *(course timetabling objective function).*

$f(y) = \_\{d_{ik}y_{ik} \mid i = 1,\ldots q;\ k = 1,\ldots p\}$

where $d_{ik}$ is the desirability of having a lecture of course $K_i$ at period k.

## 5. Approaches to Automated Timetabling

Simple, problem-specific heuristics can produce good programs, but the size and complexity of planning problems in modern universities has caused a trend towards more general problem-solving algorithms, or metaheuristics, such as annealing. simulated and scalable algorithms and taboo search. A problem-specific heuristic can be used within the framework of such an algorithm to reduce the number of possible solutions processed or to locally optimize a solution. Constraint logic programming is also a popular approach

Problem-specific inferential methods are the best way to derive good timelines. Despite the scale and complexity of the scheduling problems of the modern university Which in turn sparked a trend towards algorithms for solving more general problems that more than once encountered problems when solving them and take great time and effort to complete, or metaheuristics, such as simulated annealing and evolutionary algorithms and taboo search. Problem-specific inference can also be used in a context like this an algorithm to reduce the number of possible solutions treated, or locally improve the solution. Constraint logic programming is also a common approach.

## 5.1. Sequential Methods

These methods order events using domain heuristics and then assign the events sequentially into valid time periods so that no events in the period are in conflict with each other. In sequential methods, timetabling problems are usually represented as graphs where events are represented as vertices, while conflicts between the events are represented by edges.

For example, if some students have to attend two events there is an edge between the nodes which represent this conflict. The construction of a conflict-free timetable can therefore be modelled as a graph coloring problem. Each time period in the timetable corresponds to a color in the graph coloring problem and the vertices of a graph are colored in such a way so that no two adjacent vertices are colored by the same color

## 5.2. Cluster Methods

In these methods the set of events is split into groups which satisfy hard constraints and then the groups are assigned to time periods to fulfil the soft constraints. An early paper to describe this approach was written by White and Chan [8]Different optimization techniques have been employed to solve the problem of assigning the groups of events into time periods. The main drawback of these approaches is that the clusters of events are formed and fixed at the beginning of the algorithm and that may result in a poor-quality timetable.

Assume c courses to be examined in p examination periods. Let the c courses be the nodes ($v_1$, $v_2$,..., $V_c$) of a graph and let a student registered in both courses vi and $v_j$ be represented by an edge joining the node pair ($v_i$, $v_j$). Then the scheduling of the c courses into p periods is analogous to partitioning the nodes of the graph into no more than p disjoint sets ($s_1$, $s_2$, ...,$s_p$) such that, in any given set, there is no edge joining any of its elements. An additional aim is to find the minimum closed path through an appropriate weighted undirected graph which traverses all nodes exactly once. The solution to this problem will provide an optimum timetable for students writing examinations. This paper describes one such solution which has been used for several years at the University of Ottawa.

## 5.3. Constraint Based Approaches

In these methods a timetabling problem is modelled as a set of variables (i.e., events) to which values (i.e., resources such as rooms and time periods) have to be assigned to satisfy a number of constraints [ [6], [8]]. Usually, a number of rules is defined for assigning resources to events. When no rule is applicable to the current partial solution a backtracking is performed until a solution is found that satisfies all constraints.

## 5.4. Meta-heuristic Methods

the problem of automating timetable construction at universities is often a very challenging one. First, from a computer-science perspective most timetabling problem-formulations belong to the class of computationally NP-complete problems (see the work of Garey and Johnson [22] therefore implying that there is no known deterministic polynomially-bounded algorithm for solving them in general. Second, individual timetabling problems are also often complicated by the idiosyncratic nature of the institution and users concerned. For example, different universities will tend to have their own interpretation of what is a "feasible" and/or "good" timetable, and will therefore also tend to have their own particular set of timetable constraints that they wish to impose on their particular problem. Unfortunately, it might often be the case that an algorithmic approach that is successful for one particular problem-version may not turn out to be suitable for others.

In computing terms, timetabling problems are often modelled as Combinatorial Optimization Problems (COPs). The overall objective in a COP is to find an assignment of discrete values to variables (e.g. timeslots for each of the events that needs to be timetabled) so that that the solution is optimal according to some criteria. In other words, the problem is to find the best possible solution from all possible solutions. The techniques available for solving COPs fall into two main classes: exact algorithms and approximation algorithms. Exact algorithms are able to prove the optimality of a solution, whereas an approximation algorithm cannot.

In the case of timetabling, however, exact algorithms will typically constitute a brute-force style approach, and due to the exponential growth rates of the search spaces for these problems, their application will often only be suitable for very small problem instances. On the other hand, while approximation algorithms do not always produce optimal solutions, they do operate in polynomial time and might be able to produce solutions that are "good enough" for practical purposes. (Obviously, how often and how quickly an approximation algorithm is able to produce solutions that are "good enough" will usually be some of the criteria used to judge how effective it actually is.) we will conduct a review of the various different works that have proposed using approximation algorithms for tackling timetabling problems, paying close attention to those that have applied metaheuristic-based techniques.

Metaheuristic algorithms, which include techniques such as evolutionary algorithms, simulated annealing, tabu search, and ant colony optimization, are an important class of approximation algorithm that, over the past decade-or-so, have been applied to a variety of different COPs such as timetabling. In essence, they might be regarded as a general-purpose algorithmic framework, applicable to various COPs, with relatively few modifications generally needing to be made for each type of problem. Given this wide-ranging applicability it is arguable that they are therefore quite fitting in many areas of automated timetabling where, as we have noted, the types of constraints that are imposed will often vary from place to place.

In this thesis, we will also present arguments as to why the employment of two stages timetabling approach might sometimes be appropriate – particularly with regards to a specific university course timetabling problem-version that we will be studying in detail here.

In essence, an algorithm that uses this two-stage approach operates by first attempting to satisfy just the mandatory constraints of the problem and, assuming this task is completed successfully, will then go on to try and satisfy the remaining non-mandatory constraints, but without re-breaking the mandatory constraints in the process. Consequently, metaheuristic-based algorithms will be presented for each of these two sub-problems, and detailed analyses will be carried out in all cases [10]

# 6. Solving University Timetabling Problems

We will conduct a detailed review of the current state of this field by taking a look at some of the many different algorithms – particularly metaheuristics – that have been proposed for the various timetabling problems put forward in the literature.

## 6.1. Algorithms for University Timetabling

Given the close relationship between graph coloring and university timetabling problems, it is perhaps unsurprising that many early techniques used in timetabling algorithms were derived directly from graph coloring-based heuristics (see the survey of Carter [23]).

One early example of this sort of algorithm was provided by White and Chan [24] in 1979. This particular approach was actually used for several years at the University of Ottawa in the 1970's, and in this study, it is shown to be capable of scheduling 390 events involving 16,000 students into 25 timeslots. Basically, this method operates by first using a "largest degree first"-type heuristic to order the events.

These events are then taken one-by-one according to this ordering and are assigned to the earliest timeslot that does not cause an event-clash, with new timeslots being opened whenever necessary. Next, if the resultant solution is seen to be using more timeslots than the desired amount, the algorithm then tries to move the events in these extra timeslots into the remaining ones.

If this cannot be done, then these events are simply removed from the problem. Next, further heuristics are then used to try and find a suitable ordering for the timeslots that minimizes the soft

constraints of the problem, and finally, the events themselves are then shuffled in order to make further improvements. Another early and commonly-cited example of this sort of algorithm is the EXAMINE timetabling system documented by Carter, Laporte, and Lee in [25]. In this paper, the system – which is a backtracking sequential-assignment algorithm – is applied to a set of real-world exam timetabling problems taken from a number of different universities. (These problem instances, which were first used in this study and which are often referred to as the Carter Instances, have been publicly available for a number of years now.

They have also been used in a large number of exam timetabling papers, many of which we will look at later on in this chapter.) A number of algorithm variants are then tested and it is reported that the best performance is usually gained when two procedures are followed: first, when the events are inserted into the timetable following an ordering dictated by saturation degree heuristics (originally proposed by Brelaz [26], and which involves always selecting the node which has the highest number of colors adjacent to it); and second, when an additional algorithm is also used for trying to identify large cliques in the problem, so that the events within these cliques can then be given priority

The backtracking feature of this algorithm also enables the algorithm to undo previous assignments of events to timeslots when situations are encountered when an existing unplaced event has no feasible timeslots to which it can be assigned.

In this case, the backtracking algorithm that is used is deterministic, meaning that for a given problem instance and ordering heuristic, the same timetable will always be produced. Other approaches to timetabling problems have involved using constraint-based techniques (see the work of Deris et al. [27], Lajos [28], and Boizumault et al. [29], each of whom have applied these techniques to their own particular timetabling problems) and also integer programming (such as the algorithms of Carter [30], and Tripathy [31] in the 1980s and, more recently, Daskalaki et al. [32] in 2004).

In the past decade-or-so there has also been a large growth of interest in the application of metaheuristic-based techniques to timetabling problems. In essence, the term “metaheuristics” is used to denote a variety of stochastic-based search techniques such as simulated annealing, tabu search, iterated local search, genetic algorithm, evolutionary algorithms, and ant colony optimization.

According to the website of the Metaheuristics Network – an EU sponsored research project that was run from 2000 until 2004 [33] – a metaheuristic “… can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications [being needed] to make them adapted to a specific problem.” Given this latter characteristic, and also considering the idiosyncratic nature of timetabling problems that we have noted, it is perhaps unsurprising, therefore, that metaheuristics have become increasingly popular for addressing a number of timetabling problems in recent years.

However, when applying metaheuristic-based algorithms to timetabling problems, it is worth noting that an important aspect separating these sorts of problems from many other types of combinatorial optimization problems is the presence of both hard and soft constraints in their formulations. For any metaheuristic approach to be worth while in this case there must therefore be some sort of system that is able to deal with timetabling constraints of both types in a satisfactory way.
Our own survey of the timetabling literature with regards to this matter indicates that most metaheuristic algorithms for timetabling fall into one of three categories:

(1) One-Stage Optimization Algorithms: where a satisfaction of both the hard and soft constraints is attempted simultaneously.

(2) Two-Stage Optimization Algorithms: where a satisfaction of the soft constraints is only attempted once a feasible timetable has been found.

(3) Algorithms that allow Relaxations: Violations of the hard constraints are disallowed from the outset by relaxing some other feature of the problem. Attempts are then made to try and satisfy soft constraints, whilst also giving consideration to the task of eliminating these relaxations.

## Conclusion

Concluding this chapter, it should be clear to the reader that timetabling problems can be – and indeed have been – addressed using a variety of different computational approaches. In our review of the literature, we have devoted the majority of our discussion towards the application of metaheuristics to these problems, and have taken special note of the different ways in which these sorts of algorithm might be adapted for dealing with and distinguishing between the hard and soft constraints of a particular problem. Consequently, we have suggested that metaheuristic algorithms for timetabling can be separated into three main classes – one stage optimization algorithms, two-stage optimization algorithms, and algorithms that allow relaxations – although it is worth noting again that this classification method is not concrete, and it could well be the case that some algorithms might fit into more than one of the classes.

In the next chapter, we will conduct a survey of the field of metaheuristics and timetabling using these three categories in order to classify the various algorithms. In each subsection we will first provide a simple and general overview of the method, and will then provide a description of various works that have used this basic approach.

# CHAPTER 2 :
# GENETIC ALGORITHMS

# 1. Introduction

Charles Darwin mentioned the theory of natural evolution on the origin of species. Over several generations, living things have evolved on the basis of the natural selection principle of "survival of the fittest" to reach some wonderful tasks. The idealized shapes of albatrosses squeezing aptitude and similarity between sharks, dolphins, etc. are among the best examples of achieving random evolution over intelligence. Thus, it performs well in nature, and as a result, it should be interesting to simulate natural evolution and develop a method that solves concrete research improvement problems [11]

Whereas, with the tremendous scientific progress and the exponential acceleration witnessed by technology at the present time, new types of systems appeared, called smart systems, thus opening the way for the world to a new type of technology, called (Artificial Intelligence Technology) Technology), which is quickly developed and used in many applications today.Which greatly contributed to solving various problems faster and with impressive results. Hence, Genetic Algorithm (GA) came as a mechanism to deal with an object and make it enjoy artificial intelligence [12].

# 2. What is Genetic Algorithm?

It is one of the methods of optimization and research. This method can be classified as one of the methods of evolutionary algorithms that relies on imitating the work of nature from a Darwinian perspective. The genetic algorithm uses a search technique to find precise or approximate optimization solutions. Genetic algorithms depend mainly on the technique of natural selection or survival of the fittest. Chromosomes are used to represent the members of society, and here by society means the environment from which it will emerge, at the end of the process of evolution, the desired solution to a particular issue.

Thus, each chromosome, contains in its folds all the characteristics of the individual that it represents, so that each particular variable of the variables that represent the mathematical problem and which we seek in the end to find an optimal solution for it (the most efficient individual) It depends on several basic steps, which are Five phases are considered in a genetic algorithm :

- Initial population.
- Selection
- Fitness function
- Crossover
- Mutation

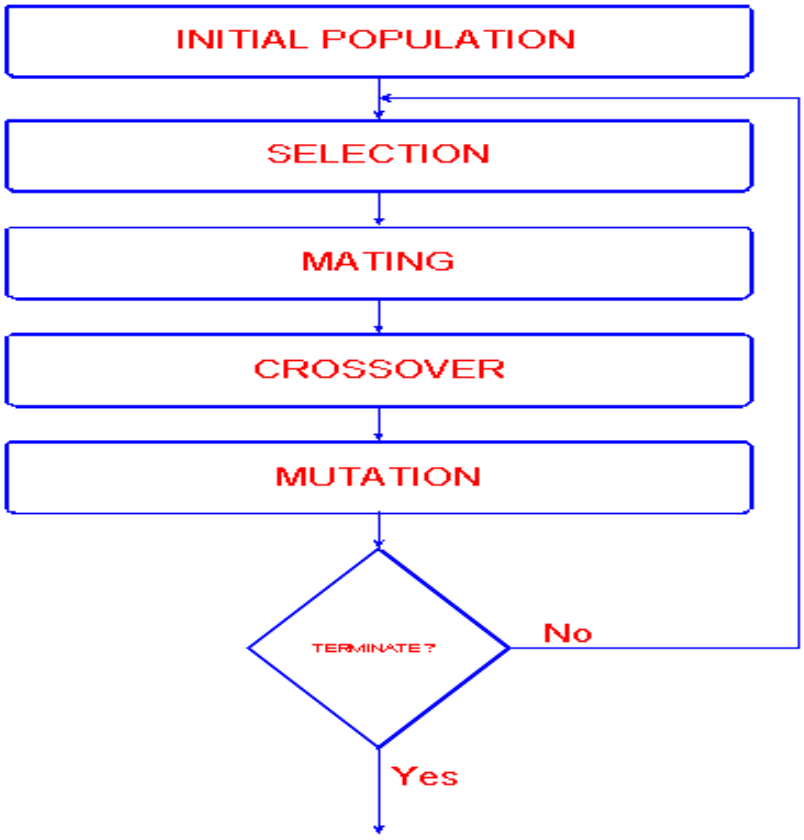



*Figure 2 Operation Of a genetic algorithm*

## 2.1. Initial population:

The first step is to create and create individuals in the population. Since the genetic algorithm is a stochastic optimization method, the genes of individuals are usually randomized [13] , as an important step in a population-based stochastic algorithm, can affect convergence velocity and quality of solutions.

Initially many of the individual solutions are randomly generated chromosomal precursors. The size of the chromosomes depends on the nature of the problem, but usually there are several hundreds or thousands of possible solutions, but it may generate a slowdown in the optimal search process. Similarly, the lack of population produces a lack of solutions received.

Then this solution may be the "classifier" in the case of reaching an optimal solution by providing optimal solutions on the one hand and on the one hand based on randomness as well that increase the likelihood of success and obtain better results.

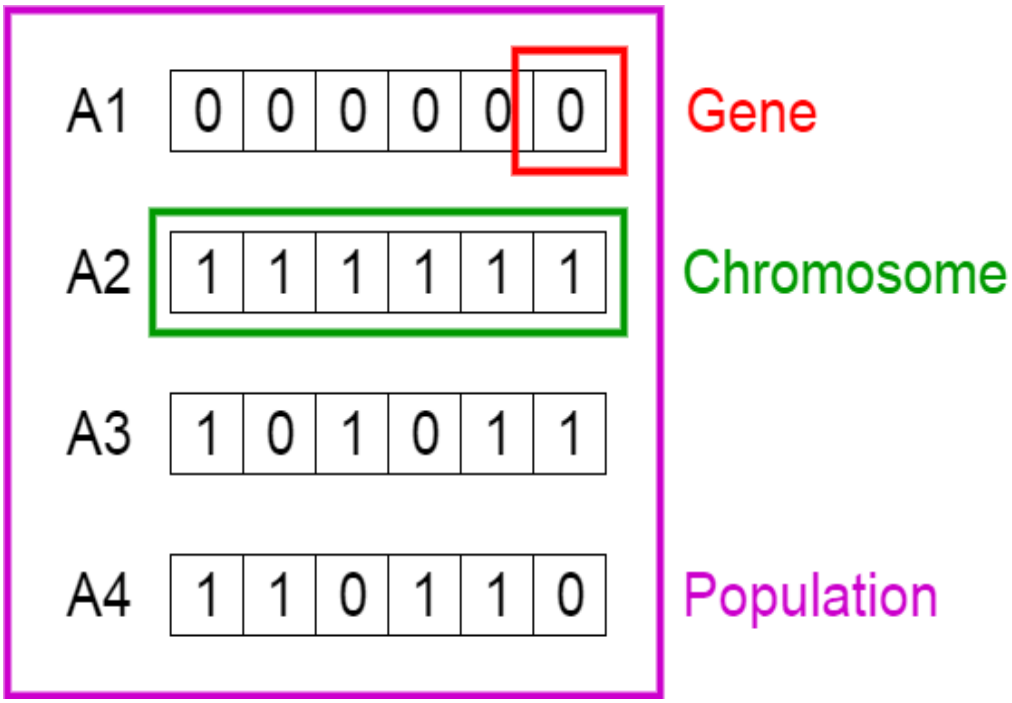



*Figure 3 Population, Chromosomes and Genes*

## 2.2. Selection:

Parental selection is the process of selecting parents who mate and reunite them to create springs for the next generation. Parental choice is critical to GA affinity as good individual parents drive better and more appropriate solutions [14] . It can increase the probability of the best answer and avoid approaching the best local solutions by using the modified selection factor and using these techniques to identify and eliminate unnecessary, inappropriate, and repetitive features that do not contribute to or reduce the accuracy of the predictive model. It also increases the likelihood of finding the best answer, and using a modified crossover factor can speed up the convergence rate, thus shortening [15] This process requires a lot of computational work, and if the number of features is large, it becomes impractical. Therefore, we need smart methods that allow the selection of features in practice [13] Among them [16]

### 2.2.1. Rank selection:

- Rank selection ranks the population first and then each chromosome gets fit from that arrangement.
- The worst will have Fitness 1, 2nd worst 2 etc., and the best will have Fitness N (number of chromosomes in the population). Enter a description of the image here
- Then all chromosomes have a chance to be selected.
- Classification based selection systems can avoid early convergence.

- But it can be mathematically expensive because it sorts the population based on their fitness value. But this method can lead to slower convergence, because the best chromosomes are not very different from the other chromosomes.

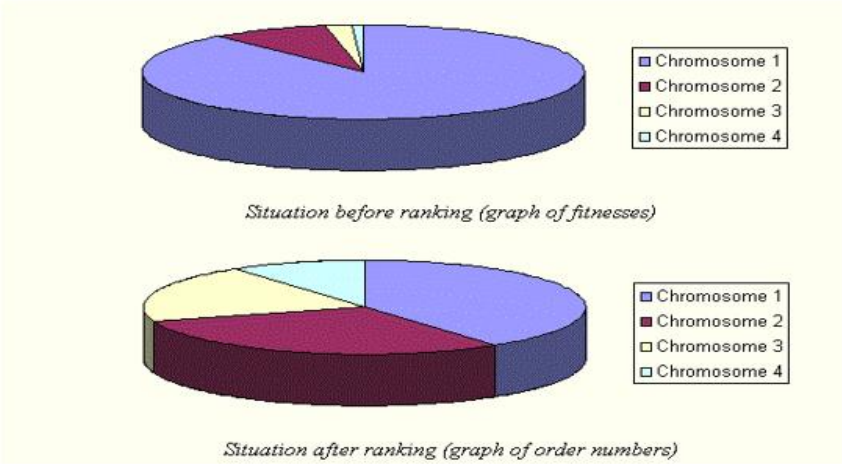



*Figure 4 Rank selection*

### 2.2.2. Tournament selection:

Course selection is a useful and robust selection mechanism commonly used by genetic algorithms (GAs). The tournament selection pressure varies directly with the tournament size - the more competitors there are, the greater the selection pressure will be created. This paper develops a model that, based on system statistics, can be used to quantitatively predict selection pressure induced by a tournament of a given size. This model is used to predict affinity rates for GA using championship selection [17].

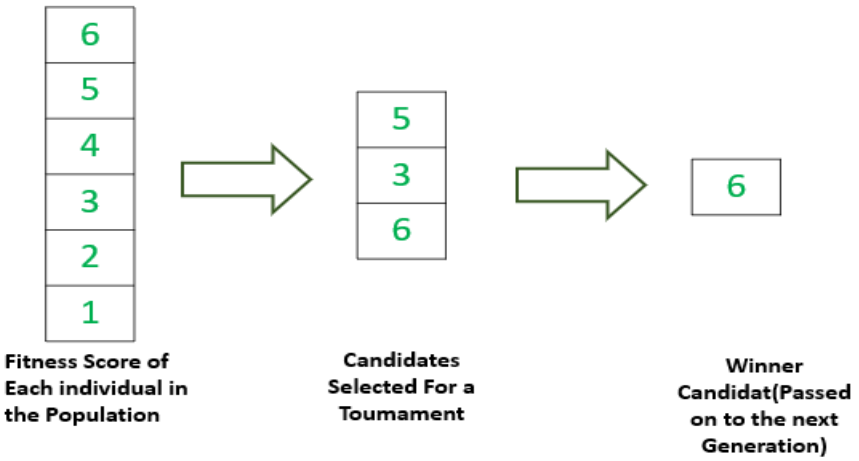



*Figure 5 Tournament selection*

### 2.2.3. Roulette wheel selection:

Parents are chosen according to their physical fitness The better the chromosomes, the higher the chances of them being selected. Imagine a roulette wheel where all the chromosomes are placed in the community, each chromosome has its large place according to its own fitness function, as in the following image description of the image center here.

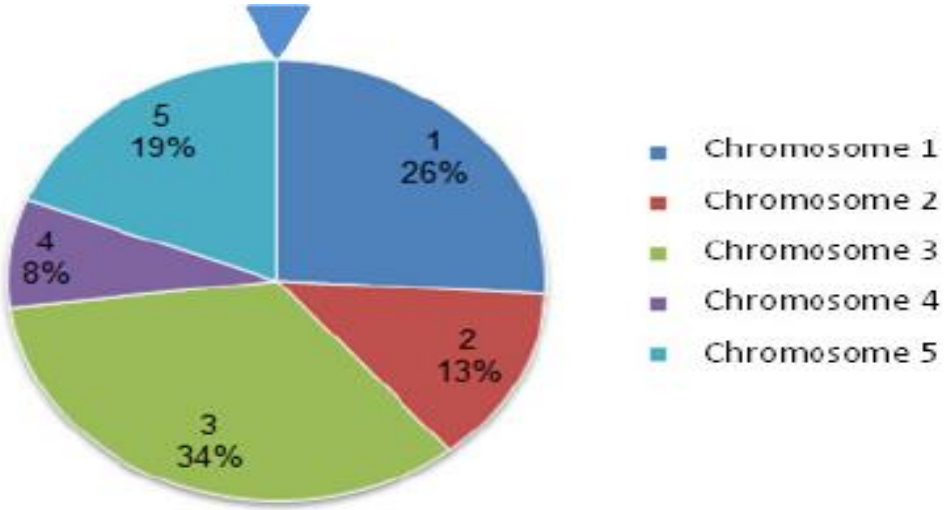



*Figure 6 Roulette wheel selection*

This choice will run into problems when dexterity varies too much. Distinguished individuals will present a bias at the start of research that may cause premature affinity and loss of diversity.

## 3. Fitness Function:

A genetic algorithm (GA) is designed to search for an optimal solution by eliminating the worst gene sequences based on fitness function [18]. The maximum fitness function can be used in multi-target genetic algorithms to obtain a variety of uncontrolled designs [19]. Finding a suitable function for a particular problem is the hardest part when it comes to formulating a problem using genetic algorithms. This result is generated by applying the fitness function to the test, or the results obtained from the solution being tested [9]

- Timetable for a week A very famous scenario where genetic algorithms can be used is the process of making timetables or timetable scheduling.
- Consider you are trying to come up with a weekly timetable for classes in a college for a particular batch. We have to arrange classes and come up with a timetable so that there are no clashes between classes. Here, our task is to search for the optimum timetable schedule.
- Since there should be no collisions among classes we should minimize the number of students having class conflicts. You can formulate the fitness function as the inverse of the number of

students with class conflicts. Lesser the number of students with class conflicts, more fit the class is.

# 4. Crossover

It is the most important stage in the genetic algorithm. For each pair of parents to be mated, a crossing point is randomly selected from within the gene pool to exchange solution bits The performance of genetic algorithms depends mainly on the type of genetic factors that include operators of crossover and mutations so that there are different cross operators and mutation to solve the problem involving a large population [20].

## 4.1. Crossover Operators

In this section we will discuss some of the most popularly used crossover operators. It is to be noted that these crossover operators are very generic and the GA Designer might choose to implement a problem-specific crossover operator as well

## 4.2. One Point Crossover

In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs.

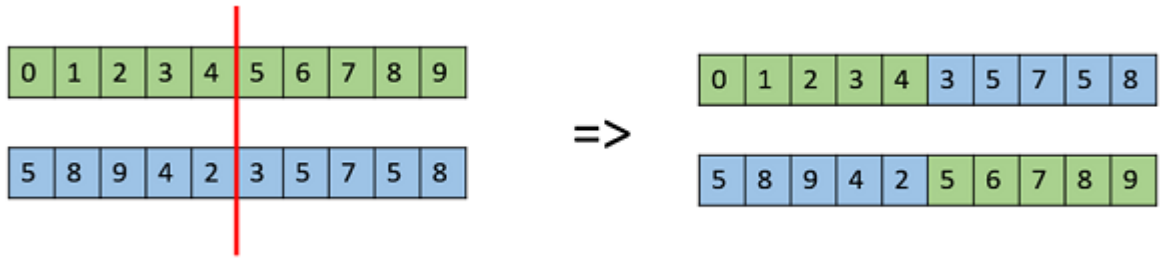



*Figure 7 One Point Crossover*

## 4.3. Multi Point Crossover

Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs.

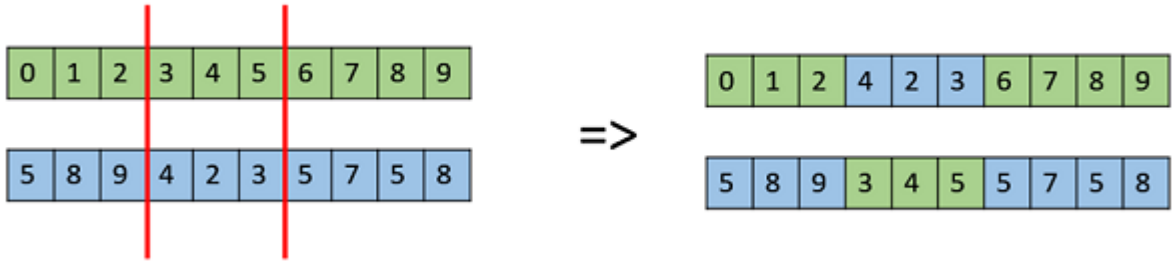



*Figure 8 Multi Point Crossover*

### 4.4. Uniform Crossover

In a uniform crossover, we don't divide the chromosome into segments, rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring. We can also bias the coin to one parent, to have more genetic material in the child from that parent.

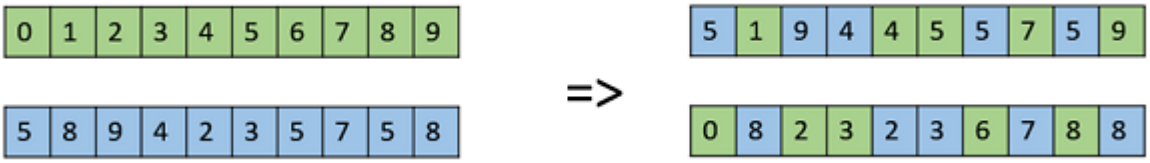



*Figure 9 Uniform Crossover*

## 5. Mutation

In simple terms, a mutation can be defined as a small random disk in the chromosome, to obtain a new solution, it is used to maintain and introduce the diversity in the genetic population, and it is usually applied with a low probability - pm. If the probability is very high, GA is reduced to random search. [21]

Mutation is seen as a background factor in maintaining genetic diversity in the population. It introduces new genetic structures in a population by randomly modifying some special building blocks ; It helps escape the trap of local minimums and maintains diversity in the population. He - she also maintains the gene pool well, thus ensuring comfort [14]

### 5.1. Mutation Operators [14]

In this section, we describe some of the most commonly used mutation operators. Like the crossover operators, this is not an exhaustive list and the GA designer might find a combination of these approaches or a problem-specific mutation operator more useful.

### 5.2. Bit Flip Mutation

In this bit flip mutation, we select one or more random bits and flip them. This is used for binary encoded GAs.

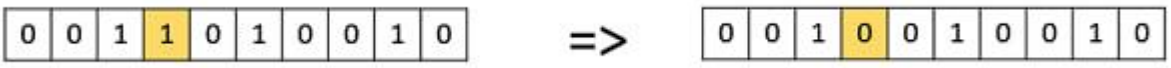



*Figure 10 Bit Flip Mutation*

### 5.3. Random Resetting

Random Resetting is an extension of the bit flip for the integer representation. In this, a random value from the set of permissible values is assigned to a randomly chosen gene.

### 5.4. Swap Mutation

In swap mutation, we select two positions on the chromosome at random, and interchange the values. This is common in permutation-based encodings.

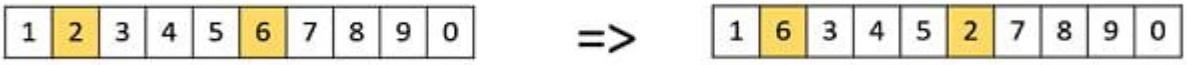



*Figure 11 Swap Mutation*

### 5.5. Scramble Mutation

Scramble mutation is also popular with permutation representations. In this, from the entire chromosome, a subset of genes is chosen and their values are scrambled or shuffled randomly.

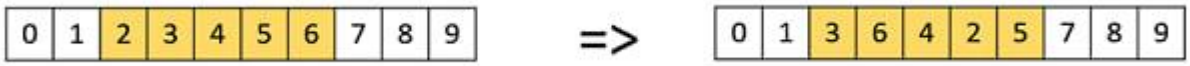



*Figure 12 Scramble Mutation*

### 5.6. Inversion Mutation

In inversion mutation, we select a subset of genes like in scramble mutation, but instead of shuffling the subset, we merely invert the entire string in the subset.

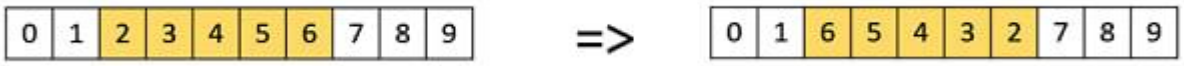



*Figure 13 Inversion Mutation*

## Conclusion

In this chapter, we have covered the working mechanism of genetic algorithms and the most important steps they go through to achieve wonderful tasks that work to give the best exact or approximate solutions to achieve optimization, from the 'step of representing solutions, chromosomes are used to represent members of the community, and here by society it means the environment from which it will emerge, because each chromosome contains all the characteristics of the individual it represents ( Figure 3), and therefore the selection of a new generation also carries qualities and solutions based on the choice of parents and diversity in the production of sons carrying diversity, after that The best solutions are selected by means of the physical fitness of genetic algorithms

Genetic algorithms are working to find the best solution to the problem of finding the best possible time, so that finalizing the interface for entering information and how to transfer and wait for the solution remains what we will cover in the next chapter.

# CHAPITRE 3 :
# APIs
## (APPLICATION PROGRAMMING INTERFACES)

# 1. Introduction

The individual in his various tasks in our time depends on solving his problems through applications, which have taken up a large space in our lives, but the hour and place we need them differ, and also the capacity of programs and applications varies, especially if they contain a large bdd and can be updated as well. From this point of view, an application programming interface (API) has been prepared, which is a way to programmatically interact with a separate software component or resource, meaning that it gives an interface to receive orders or requests to go in turn and verify your order and the restaurant server is the best example of this.

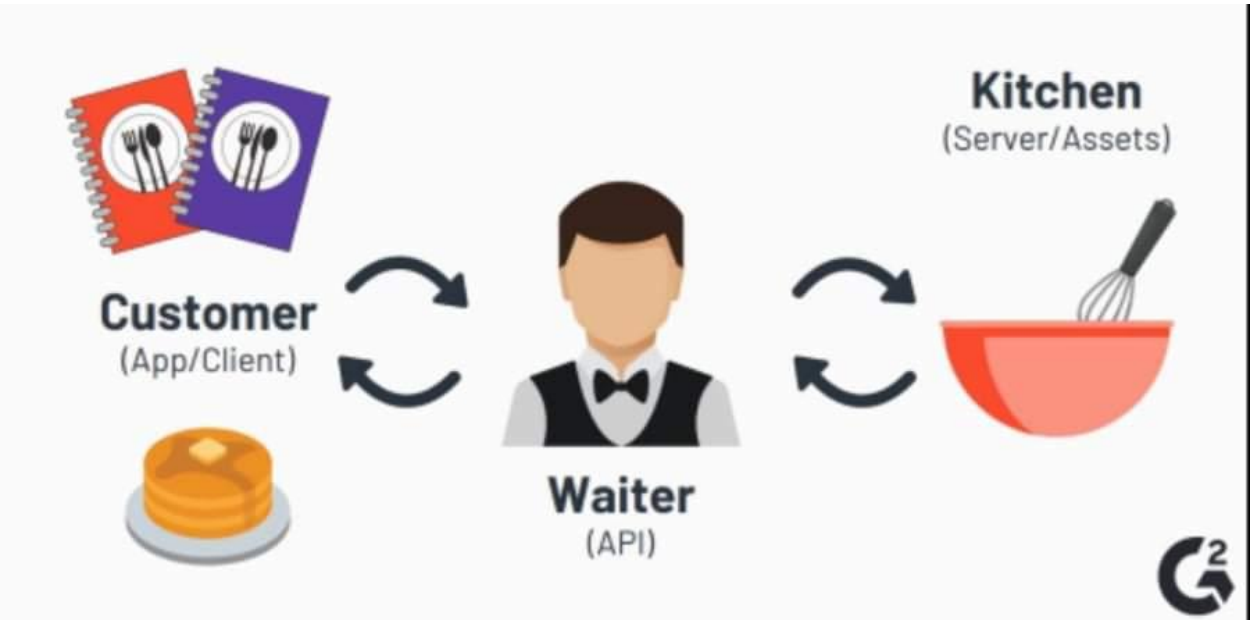



*Figure 14 example  Application Programming Interface*

API stands for Application Programming Interface, a concept that applies everywhere from command line tools to enterprise Java code to Ruby on Rails web applications. [*34*] Unless you write every line of code from scratch, it will interact with external software components, each with its own API, even if you write something entirely from scratch, a well-designed software application will have internal APIs to help organize the code and make components More reusable [*34*].

## 2 . What is an API?

An **API** is a set of programming code that enables data transmission between one software product and another. It also contains the terms of this data exchange.

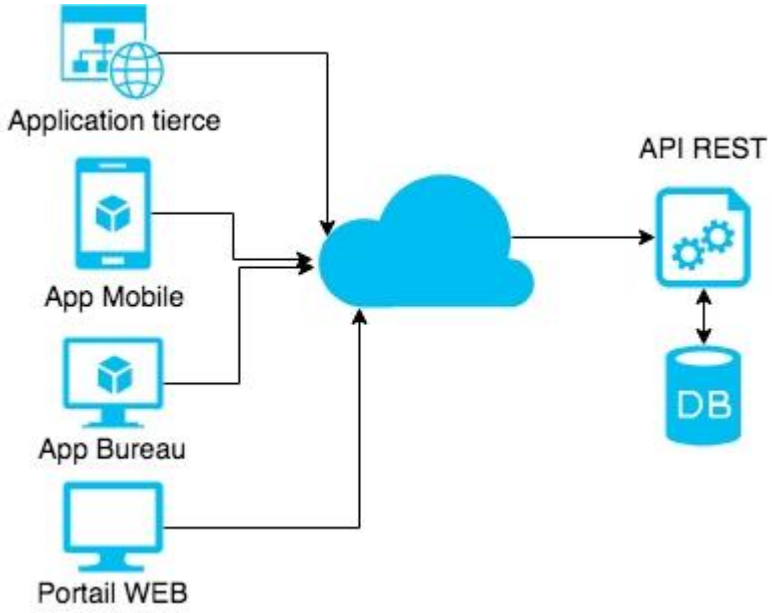



*Figure 15 Application Programming Interface*

Application programming interfaces consist of two components:

- Technical specification describing the data exchange options between solutions with the specification done in the form of a request for processing and data delivery protocols
- Software interface written to the specification that represents it

The software that needs to access information or functionality from another software, calls its API while specifying the requirements of how data/functionality must be provided. The other software returns data/functionality requested by the former application. And the interface by which these two applications communicate is what the API specifies. The Red Hat specialists note that APIs are sometimes considered contracts, where documentation is an agreement between the parties: “If party first sends a remote request structured a particular way, this is how the second party’s software will respond.” The API documentation is a manual for developers that includes all necessary information on how to work with the API and use the services it provides. Each API contains and is implemented by function calls – language statements that request software to perform particular actions and services. Function calls are phrases composed of verbs and nouns, for example:

- Start or finish a session
- Get amenities for a single room type
- Restore or retrieve objects from a server.

Function calls are described in the API documentation.

APIs serve numerous purposes. Generally, they can simplify and speed up software development. Developers can add functionality (i.e., recommender engine, accommodation booking, image recognition, payment processing) from other providers to existing solutions or build new applications using services by third-party providers. In all these cases, specialists don't have to deal with source code, trying to understand how the other solution works. They simply connect their software to another one. In other words, APIs serve as an abstraction layer between two systems, hiding the complexity and working details of the latter.

# 3. Types of APIs

## 3.1. Types of APIs by availability

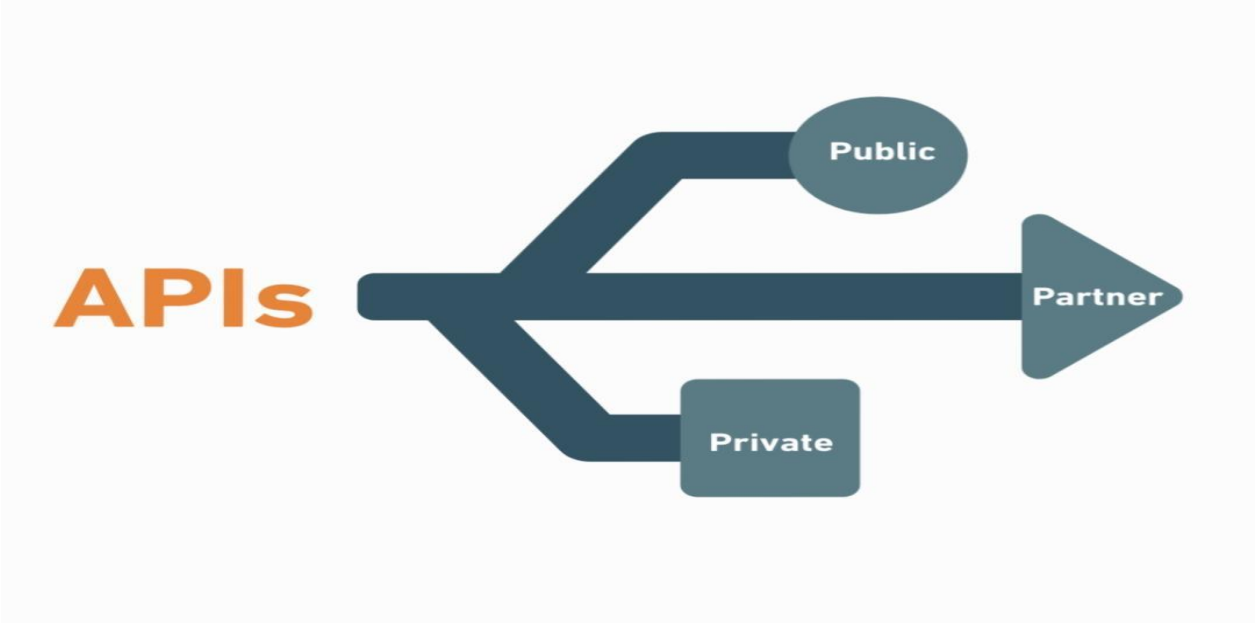



*Figure 16 Types of APIs by availability*

**Private APIs:** These application software interfaces are designed for improving solutions and services within an organization. In-house developers or contractors may use these APIs to integrate a company's IT systems or applications, build new systems or customer-facing apps leveraging existing systems. Even if apps are publicly available, the interface itself remains available only for those working directly with the API publisher. The private strategy allows a company to fully control the API usage. [34]

**Partner APIs:** Partner APIs are openly promoted but shared with business partners who have signed an agreement with the publisher. The common use case for partner APIs is software integration between two parties. A company that grants partners with access to data or capability benefits from extra revenue streams. At the same time, it can monitor how the exposed digital assets are used, ensure whether third-party solutions using their APIs provide decent user experience, and maintain corporate identity in their apps. [34]

**Public APIs :** Also known as developer-facing or external, these APIs are available for any third-party developers. A public API program allows for increasing brand awareness and receiving an additional source of income when properly executed. [34]

There are two types of public APIs – open (free of charge) and commercial ones. The Open API Definition suggests that all features of such an API are public and can be used without restrictive terms and conditions. For instance, it's possible to build an application that utilizes the API without explicit approval from the API supplier or mandatory licensing fees. The definition also states that the API description and any related documentation must be openly available, and that the API can be freely used to create and test applications. [34]

Commercial API users pay subscription fees or use APIs on a pay-as-you-go basis. A popular approach among publishers is to offer free trials, so users can evaluate APIs before purchasing subscriptions.

## 3.2. APIs by use cases

APIs can be classified according to the systems for which they are designed

**Database APIs :** Database APIs enable communication between an application and a database management system. Developers work with databases by writing queries to access data, change tables, etc. The Drupal 7 Database API, for example, allows users to write unified queries for different databases, both proprietary and open source (Oracle, MongoDB, PostgreSQL, MySQL, CouchDB, and MSSQL). Another example is ORDS database API, which is embedded into Oracle REST Data Services.

**Operating systems APIs :** This group of APIs defines how applications use the resources and services of operating systems. Every OS has its set of APIs, for instance, Windows API or Linux API (kernel–user space API and kernel internal API). Apple provides API reference for macOS and iOS in its developer documentation. APIs for building applications for Apple's macOS desktop operating system are included in the Cocoa set of developer tools. Those building apps for the iOS mobile operating system use Cocoa Touch – a modified version of Cocoa.

**Remote APIs:** Remote APIs define standards of interaction for applications running on different machines. In other words, one software product accesses resources located outside the device that requests them, which explains the name. Since two remotely located applications are connected over a communications network, particularly the internet, most remote APIs are written based on web standards. Java Database Connectivity API and Java Remote Method Invocation API are two examples of remote application programming interfaces.

**Web APIs:** This API class is the most common. Web APIs provide machine-readable data and functionality transfer between web-based systems which represent client-server architecture. These APIs mainly deliver requests from web applications and responses from servers using Hypertext Transfer Protocol (HTTP). Developers can use web APIs to extend the functionality of their apps or sites. For instance, the Pinterest API comes with tools for adding users' Pinterest data like boards or Pins to a website. Google Maps API enables the addition of a map with an organization's location.

# 4. API specifications/protocols

The goal of API specifications is to standardize data exchange between web services. In this case, standardization means the ability of diverse systems, written in different programming languages and/or running on different OSs, or using different technologies, to seamlessly communicate with each other.

## 4.1 Remote Procedure Call (RPC)

Web APIs may adhere to resource exchange principles based on a Remote Procedure Call. This protocol specifies the interaction between client-server-based applications. One program (client) requests data or functionality from another program (server), located in another computer on a network, and the server sends the required response. RPC is also known as a subroutine or function call. One of two ways to implement a remote procedure call is SOAP.

## 4.2 Service Object Access Protocol (SOAP)

SOAP is a lightweight protocol for exchanging structured information in a decentralized, distributed environment, according to the definition by Microsoft that developed it. Generally speaking, this specification contains the syntax rules for request and response messages sent by
web applications. APIs that comply with the principles of SOAP enable XML messaging between systems through HTTP or Simple Mail Transfer Protocol (SMTP) for transferring mail.

Extensible markup language (XML) is a simple and very flexible text format widely used for data storage and exchange over the internet or other networks. XML defines a set of rules for encoding documents in a format that both humans and machines can read. The markup language is a collection of symbols that can be placed in the text to delineate and label the parts of the text document. XML text documents contain self-descriptive tags of data objects, which makes them easily readable. SOAP is

mostly used with enterprise web-based software to ensure high security of transmitted data. SOAP APIs are preferred among providers of payment gateways, identity management and CRM solutions, as well as financial and telecommunication services. PayPal's public API is one of the commonly known SOAP APIs. It's also frequently used for legacy system support.

### 4.3 Representational State Transfer (REST)

The term *REST* was introduced by computer scientist Roy Fielding in a dissertation in 2000. Unlike SOAP, which is a protocol, REST is a software architectural style with six constraints for building applications that work over HTTP, often web services. The World Wide Web is the most common realization and application of this architecture style.

REST is considered a simpler alternative to SOAP, which many developers find difficult to use because it requires writing a lot of code to complete every task and following the XML structure for every message sent. REST follows another logic since it makes data available as resources. Each resource is represented by a unique URL, and one can request this resource by providing its URL. Web APIs that comply with REST architectural constraints are called RESTful APIs. These APIs use HTTP requests (AKA methods or verbs) to work with resources: GET, PUT, HEAD, POST, PATCH, CONNECT, TRACE, OPTIONS and DELETE.

RESTful systems support messaging in different formats, such as plain text, HTML, YAML, XML, and JSON, while SOAP only allows XML. The ability to support multiple formats for storing and exchanging data is one of the reasons REST is a prevailing choice for building public APIs these days.Social media giants and travel companies provide external APIs to improve their brand visibility even more. Twitter has numerous RESTful APIs; Expedia has both SOAP and RESTful APIs for its partners. If you consider redefining your travel and hospitality business offering, dive deep into the world of travel and booking APIs with our dedicated article.

**JavaScript Object Notation (JSON)** is a lightweight and easy-to-parse text format for data exchange. Its syntax is based on a subset of the Standard ECMA-262 3rd Edition. Each JSON file contains collections of name/value pairs and ordered lists of values. Since these are universal data structures, the format can be used with any programming language. JSON has been widely adopted thanks to the popularity of REST.

## 5. API documentation

No matter how many opportunities for creating or extending software products API gives, it would remain an unusable piece of code if developers didn't understand how to work with it. Well-written and structured API documentation that explains how to effectively use and integrate an API in an easy-to-comprehend manner will make a developer happy and eager to recommend the API to

peers.The API documentation is a reference manual with all needed information about the API, including functions, classes, return types, and arguments. [34]

Numerous content elements make good documentation, such as:

- a quick start guide
- authentication information
- explanations for every API call (request)
- examples of every request and return with a response description, error messages, etc.
- samples of code for popular programmatic languages like Python, Java, JavaScript, or PHP;
- tutorials
- SDK examples (if SDKs are available) illustrating how to access the resource, etc.

Documentation may be *static* and *interactive.* The latter allows for trying out APIs and see return results and usually consists of two columns: human and machine. The human column contains API descriptions, and the machine one has a console to make calls and contains info that clients and servers will be interested in when testing the API.Generation is the process of documenting APIs by developers and technical writers. The specialists may use API documentation solutions (i.e., Swagger tools, Postman, Slate, or ReDoc) to unify documentation structure and design.

## Conclusion

The role of APIs is considerably greater if we look at it not only from the software development angle but also from the business collaboration angle. These machine-readable interfaces for resource exchange are like delivery services that work under the hood and enable that needed technological connectivity. More than 60 percent of the participants in the Current State of API Integration 2018 report agreed that API integration is critical to their business strategy. The study also suggested over 50 percent of all businesses would partner via APIs.

In this regard, the two main tasks for decision makers and developers are to select the API that works for a company’s specific business needs and understand how to effectively use it.

# CHAPITRE 4 :
# IMPLEMENTATION OF THE SOLUTION TO TIME TABLING

# 1. Introduction

In this chapter we will explain our approach to solve automated time tabling use genetic algorithm as a meta heuristic what is the best time tabling?

In our problem we have the soft constraints and hard constraints so, our solution is based on the work of researchers on genetic algorithms of the last decades After that , we will implement a Application in the following section we will describe and enlarge the solution based on genetic algorithms

# 2. UML language

Unified Modeling Language (UML) is a pictogram-based graphical modeling language designed to provide a standardized method for visualizing the design of a system. It is commonly used in software development and object-oriented design.

UML allows you to define and visualize a model, using diagrams. A UML diagram is a graphical representation, which is concerned with a specific aspect of the model, and each type of diagram has a structure that conveys a precise semantics and the combination of the different types of diagrams provides a complete view of the static and dynamic aspects of the diagram. 'a system.

# 3. Analysis

The analysis phase makes it possible to list the expected results, in terms of functionalities, performance, robustness, maintenance, security, extensibility, etc.

## 3.1 Sequence diagrams

The sequence diagram makes it possible to show the interactions of objects within the framework of a scenario of a diagram of the use cases The goal being to describe how the actions between the actors or objects take place.

## Sequence diagrams

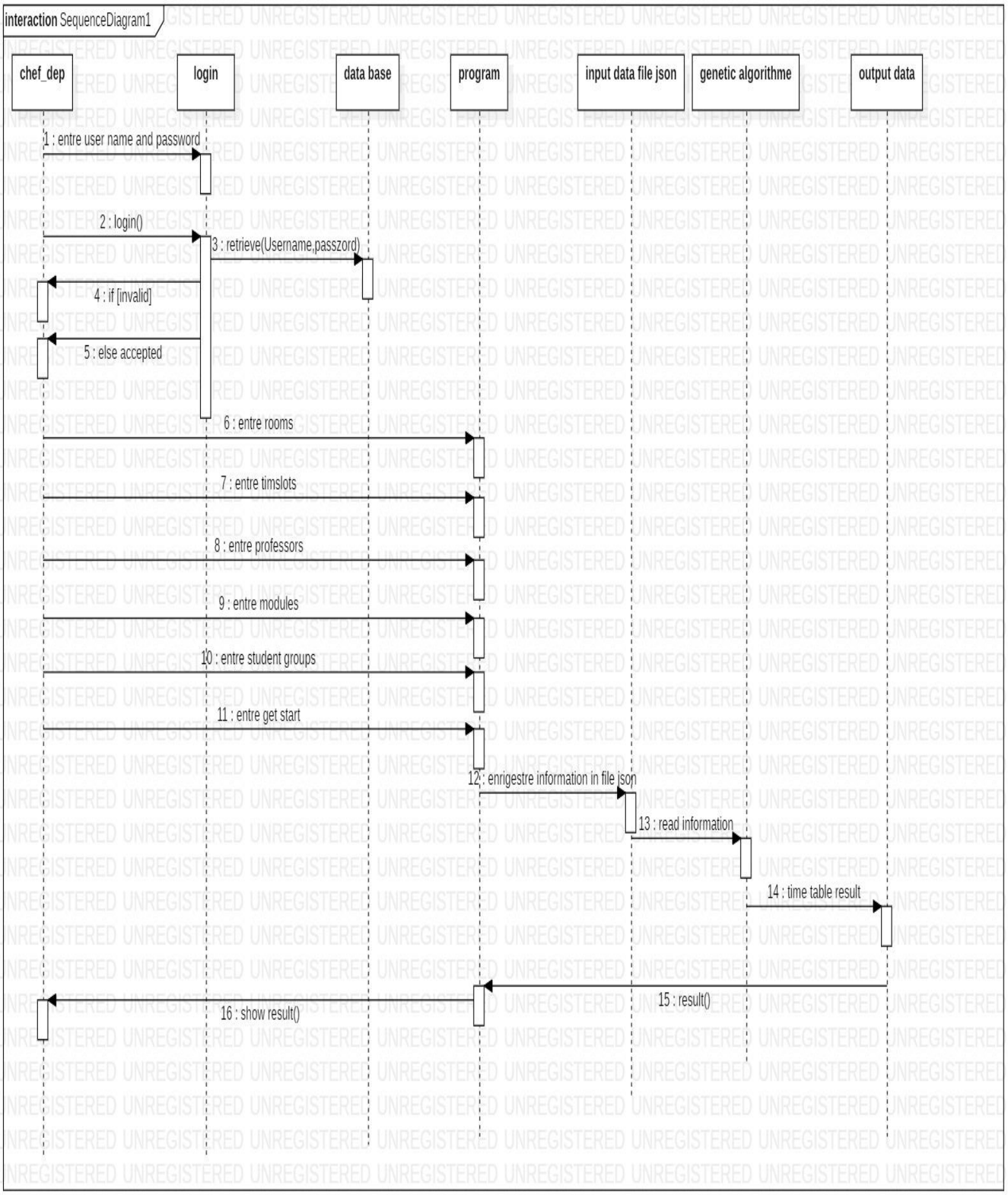



*Figure 17 Sequence diagrams*

## 3.2 Class diagram:

It represents (Figure 18) the classes involved in the system. The class diagram is a static representation of the elements that make up a system and their relationships. Each application that will implement the system will be an instance of the different classes that make it up. As such, it will be necessary to keep in mind that a class is a model and the object its realization.

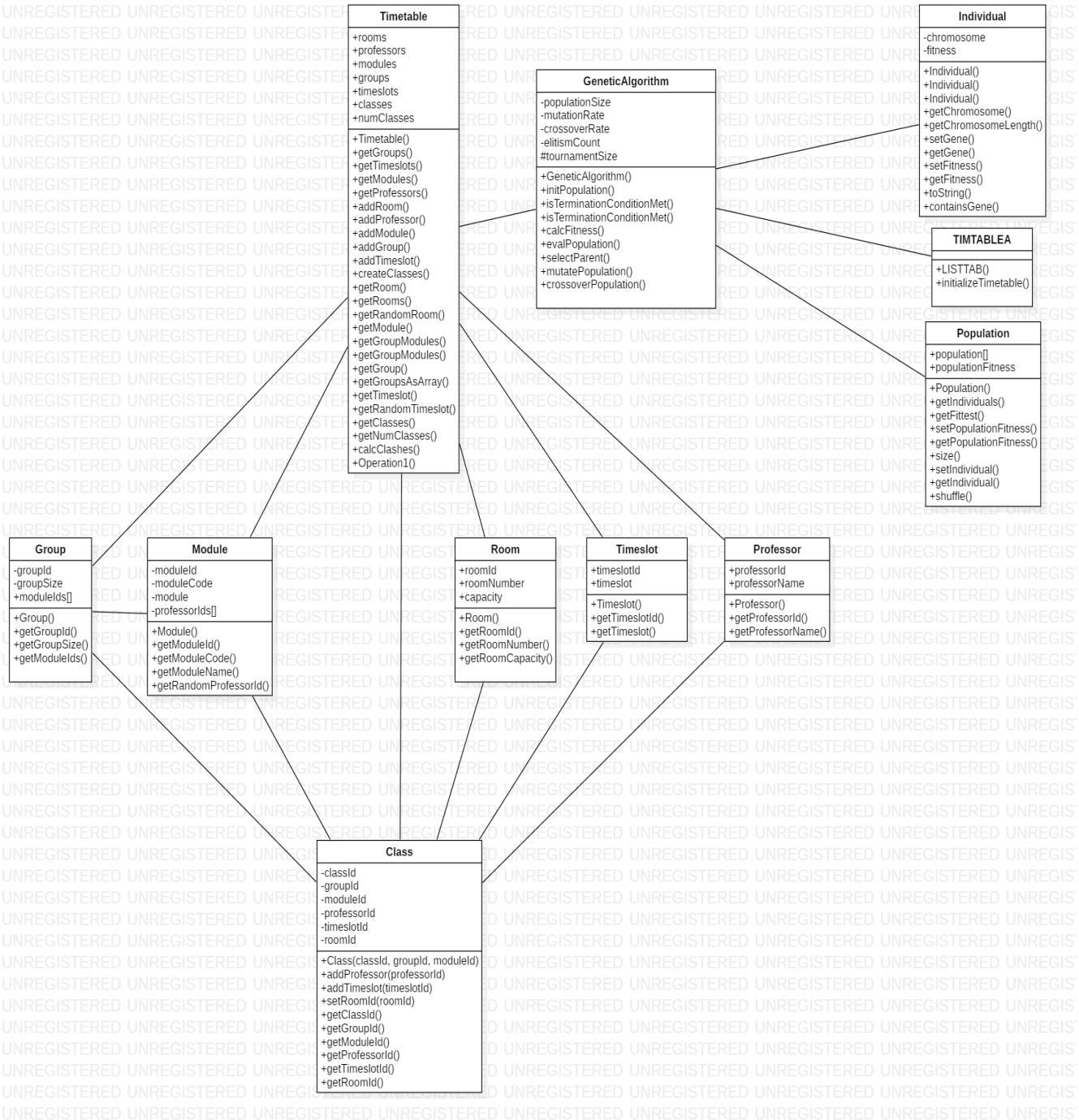



*Figure 18 Class diagram*

- Class

  A simple class abstraction -- basically a container for class, group, module, professor, timeslot, and room IDs

  Methods:

  - Initialize new Class
  - Add professor to class
  - Add timeslot to class
  - Add room to class
  - Get classId
  - Get groupId
  - Get moduleId
  - Get professorId
  - Get timeslotId
  - Get roomId

- Group

A simple "group-of-students" abstraction. Defines the modules that the group is enrolled in

  - Initialize Group
  - Get groupId
  - Get groupSize
  - Get array of group's moduleIds

- Module

Simple course module abstraction, which defines the Professors teaching the module

  - Initialize new Module
  - Get moduleId
  - Get module name
  - Get random professor Id

- Professor

  Simple Professor abstraction

  - Initalize new Professor
  - Get professorId
  - Get professor's name

- Room

Simple Room abstraction -- used to store the room capacity and compare against the student Group's size.

- Initialize new Room
- Get Room Id
- Get Room Number
- Get Room Capacity

- Timeslot

  Simple timeslot abstraction -- just represents a timeslot (like "Wed 9:00am-11:00am")
  - getTimeslotId
  - getTimeslot
- Timetable

Timetable is the main evaluation class for the class scheduler Genetic Algorithm.A timetable represents a potential solution in human-readable form, unlike an Individual or a chromosome. This timetable class, then, can read a chromosome and develop a timetable from it, and ultimately can evaluate the timetable for its fitness and number of scheduling clashes. The most important methods in this class are createClasses and calcClashes.

The createClasses method accepts an Individual (really, a chromosome), unpacks its chromosome, and creates Class objects from the genetic information. Class objects are lightweight; they're just containers for information with getters and setters, but it's more convenient to work with them than with the chromosome directly.

The calcClashes method is used by GeneticAlgorithm.calcFitness, and requires that createClasses has been run first. calcClashes looks at the Class objects created by createClasses, and figures out how many hard constraints have been violated. Create classes using individual's chromosome One of the two important methods in this class; given a chromosome, unpack it and turn it into an array of Class (with a capital C) objects. These Class objects will later be evaluated by the calcClashes method, which will loop through the Classes and calculate the number of conflicting timeslots, rooms, professors, etc.

While this method is important, it's not really difficult or confusing. Just loop through the chromosome and create Class objects and store them.

Calculate the number of clashes between Classes generated by a chromosome.

The most important method in this class; look at a candidate timetable and figure out how many constraints are violated.

Running this method requires that create Classes has been run first (in order to populate this. Classes). The return value of this method is simply the number of constraint violations (conflicting professors timeslots, or rooms), and that return value is used by the GeneticAlgorithm.calcFitness method.

There's nothing too difficult here either -- loop through this. Classes, and check constraints against the rest of the this. Classes. The two inner `for` loops can be combined here as an optimization, but kept separate for clarity. For small values of this. classes. Length it doesn't make a difference, but for larger values it certainly does.

- Individual

In this case, the chromosome is an array of integers rather than a string

Initializes random individual based on a timetable

The Timetable class is a bit overloaded. It knows both fixed information (the courses that MUST be scheduled, the professors that MUST be given jobs, the classrooms that DO exist) -- but it also understands how to parse and unpack chromosomes which contain variable information (which professor teaches which class and when?)

In this case, we use the Timetable for the fixed information only, and generate a random chromosome, making guesses at the variable information.

Given the fixed information in a Timetable, we create a chromosome that randomly assigns timeslots, rooms, and professors to the chromosome for each student group and module.

- Population
  - Initializes blank population of individuals
  - The size of the population
  - The length of the individuals chromosome
  - Initial population
  - Loop over population size
  - Create individual
  - Add individual to population
  - Get individuals from the population
  - Find fittest individual in the population
  - Order population by fitness
  - The population's total fitness
  - Get individual at offset
  - Shuffles the population in-place
- Genetic Algorithm
  - Initialize population

- The length of the individuals chromosome
- Check if population has met termination condition
- Create new timetable object to use -- cloned from an existing timetable
- Calculate fitness
- Loop over population evaluating individuals and summing population
- Selects parent for crossover using tournament selection
- Tournament selection works by choosing N random individuals, and then
- choosing the best of those.
- Create tournament
- Add random individuals to the tournament
- Apply mutation to population
- Initialize new population
- Loop over current population by fitness
- Create random individual to swap genes with
- Loop over individual's genes
- Skip mutation if this is an elite individual
- Does this gene need mutation?
- Swap for new gene
- Add individual to population
- Create new population
- Loop over current population by fitness
- Apply crossover to this individual?
- Initialize offspring

Find second parent

- Loop over genome
- Use half of parent1's genes and half of parent2's genes
- Add offspring to new population
- Add individual to new population without applying crossover

- TimetableGA

Don't be daunted by the number of classes in this chapter -- most of them are just simple containers for information, and only have a handful of properties with setters and getters.

The real stuff happens in the Genetic Algorithm class and the Timetable class. The Timetable class is what the genetic algorithm is expected to create a valid version of -- meaning, after all is said and done, a chromosome is read into a Timetable class, and the Timetable class creates a nicer, neater

representation of the chromosome by turning it into a proper list of Classes with rooms and professors and whatnot. The Timetable class also understands the problem's Hard Constraints (ie, a professor can't be in two places simultaneously, or a room can't be used by two classes simultaneously), and so is used by the Genetic Algorithm's calcFitness class as well. Finally, we overload the Timetable class by entrusting it with the "database information" generated here in initialize Timetable.

Normally, that information about what professors is employed and which classrooms the university has would come from a database, but this isn't a book about databases so we hardcode it.

# 4.The different tools used:

## 4.1 NetBeans 8.2

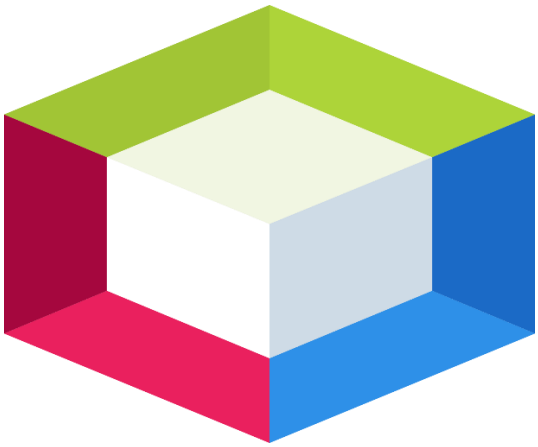


*Figure 19 Apache NetBeans Logo*

NetBeans is an integrated development environment (IDE) for Java, placed in open source by Sun. In addition to Java, NetBeans also provides support for various other languages, such as C, C ++, XML and HTML. It includes all the characteristics of a modern IDE (color editing, multi-language projects, graphic interface and web page editor).

NetBeans is available on Windows, Linux, and other operating systems. It is itself developed in Java, which can make it quite slow and consuming in memory resources.

## 4.2 WAMP SERVER

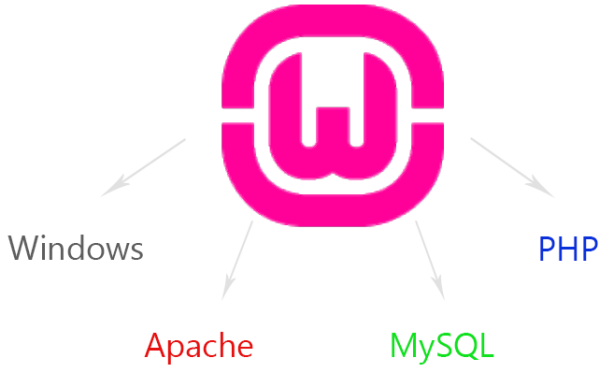



*Figure 20 wamp server*

WAMP SERVER accessible in local network

WampServer is a Windows-based web development platform for dynamic web applications using Apache2 server, PHP scripting language, and MySQL database. It also has PHPMyAdmin to more easily manage your databases

## 4.3 MySql


*Figure 21 Mysql logo*

MySQL Data base Service is a fully managed database service for deploying cloud native applications using the world's most popular open source database. This service is 100% developed, managed and supported by the MySQL team.

## 4.4 JSON

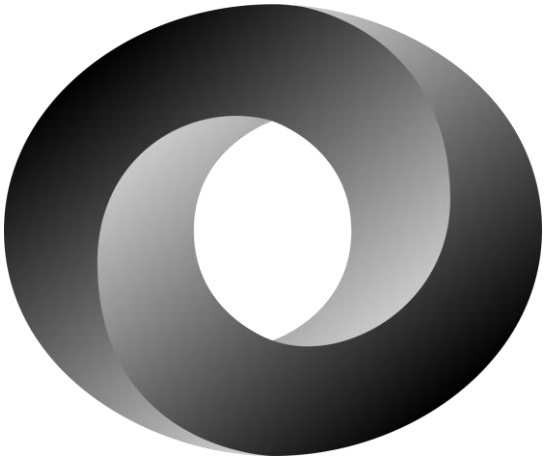


*Figure 22 JSON LOGO*

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999.

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

## 4.5 Jtatoo

*Figure 23 Jtatoo logo*

JTattoo consists of several different Look and Feels for Swing applications. All of them enables developers to improve their application with an excellent user interface. So JTattoo opens desktop applications the door to end users who are unfortunate with the Look and Feels shipped with the standard JDK.

## 5. System Presentation


*Figure 24 login*

You must enter Username , Password and Clicking the Login Button

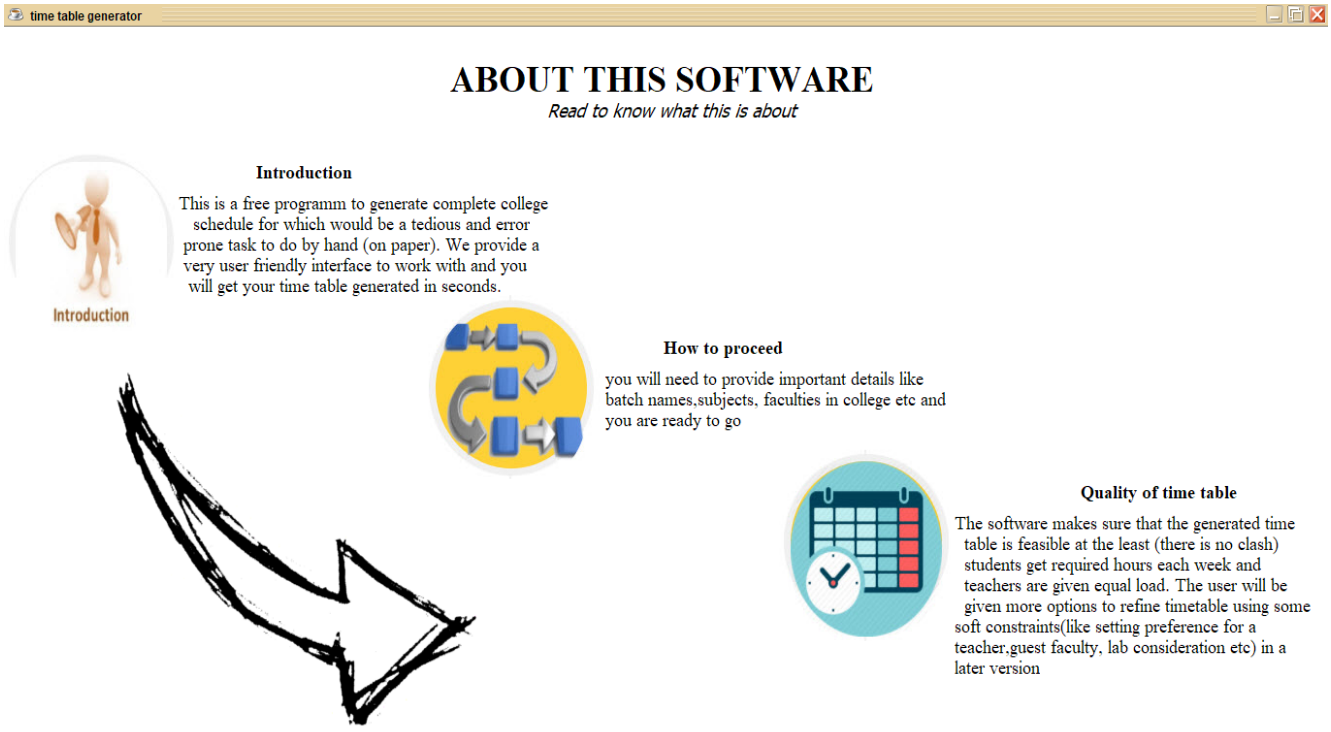



*Figure 25 The main interface of the program*

**Introduction**

This is a free program to generate complete college schedule for which would be a tedious and error prone task to do by hand (on paper). We provide a very user-friendly interface to work with and you will get your time table generated in seconds.

**How to proceed**

you will need to provide important details like batch names, subjects, faculties in college etc. and you are ready to go

**Quality of time table**

The software makes sure that the generated time table is feasible at the least (there is no clash) students get required hours each week and teachers are given equal load. The user will be given more options to refine timetable using some soft constraints (like setting preference for a teacher, guest faculty, lab consideration etc.)

- Clic King the Arrow Button

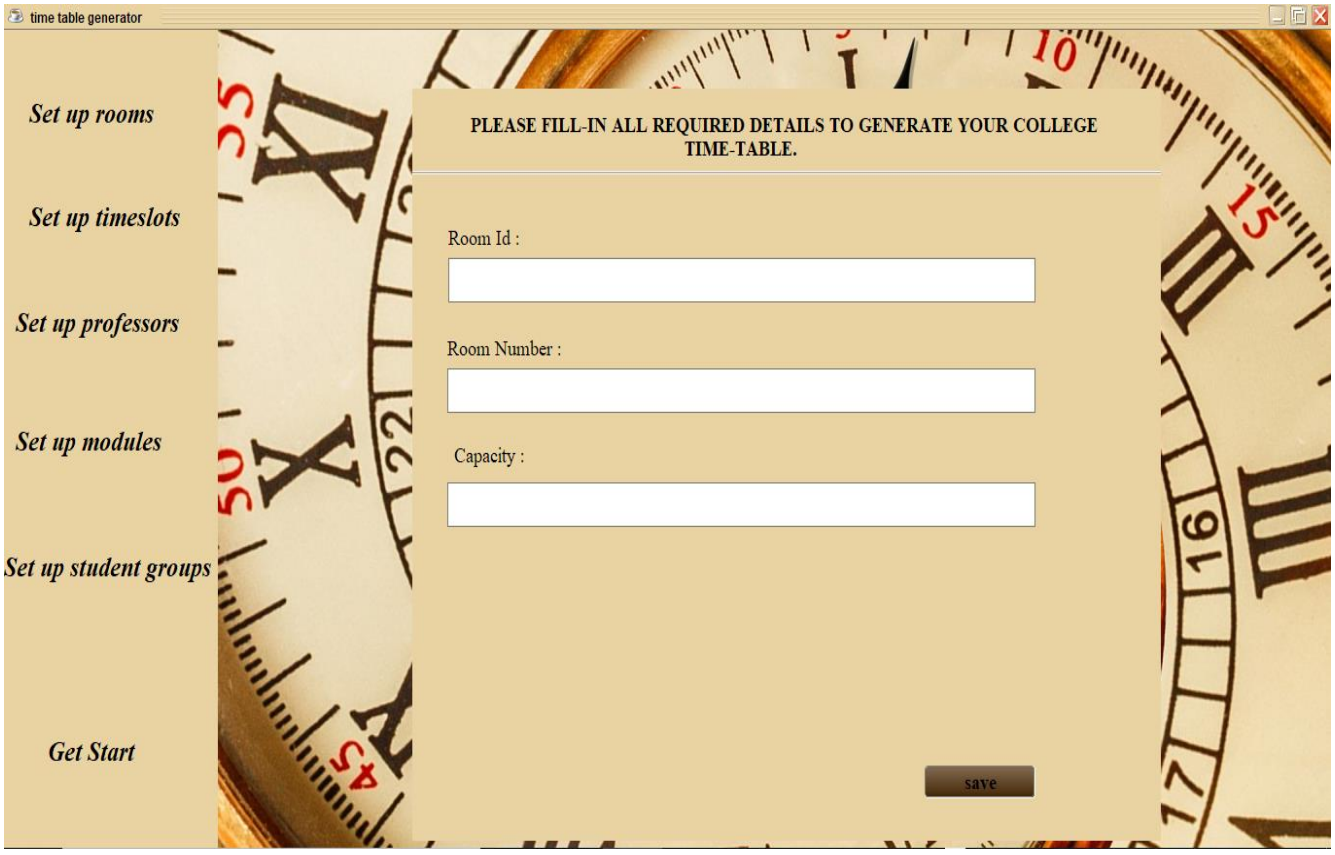



*Figure 26 Set up rooms page*

Fill in the information and for each room

1.RoomID

2.Room Number

3.Capacity

Clicking the save Button when you have finished all the rooms go to set up timeslots

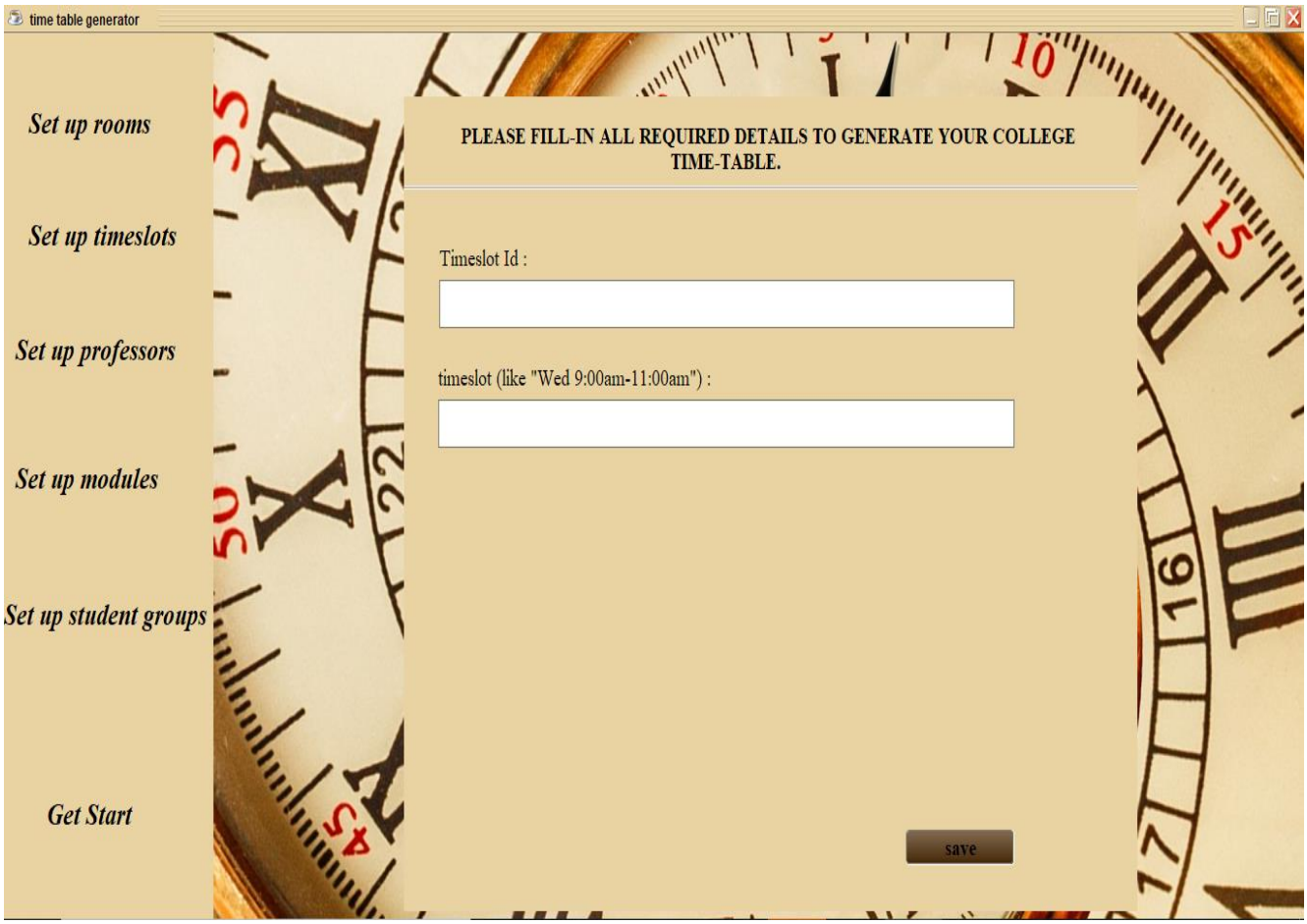



*Figure 27 Set up timeslots page*

Fill in the information and for each timeslot of the day

1. Timeslot Id
2. Timeslot (like "Wed 9:00am-11:00am") :

Clicking the save Button when you have finished all the timeslots of the days go to set up Professor

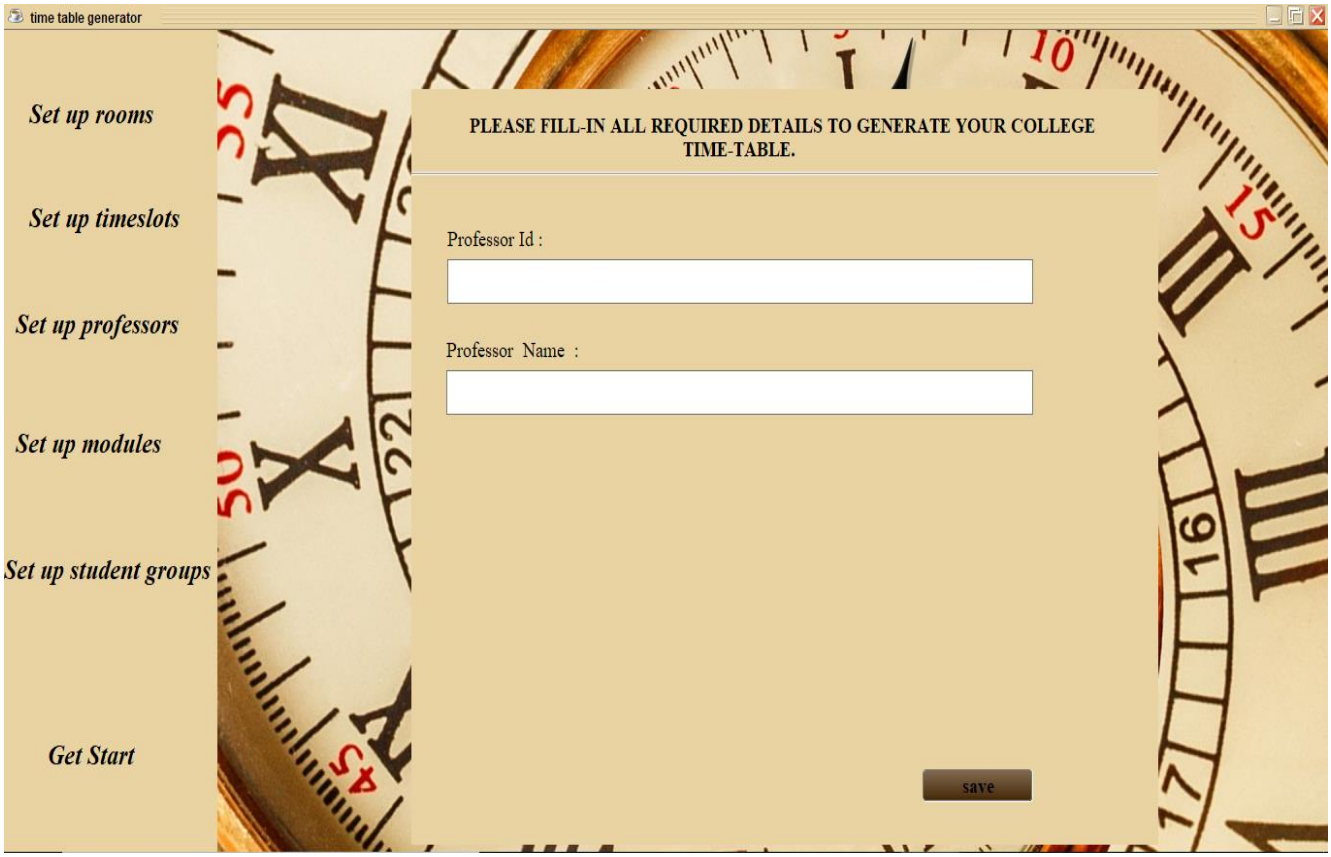



*Figure 28 set up professors page*

Fill in the information and for each professor

1. professor Id
2. professor Name

Clicking the save Button when you have finished all the professors go to set up modules

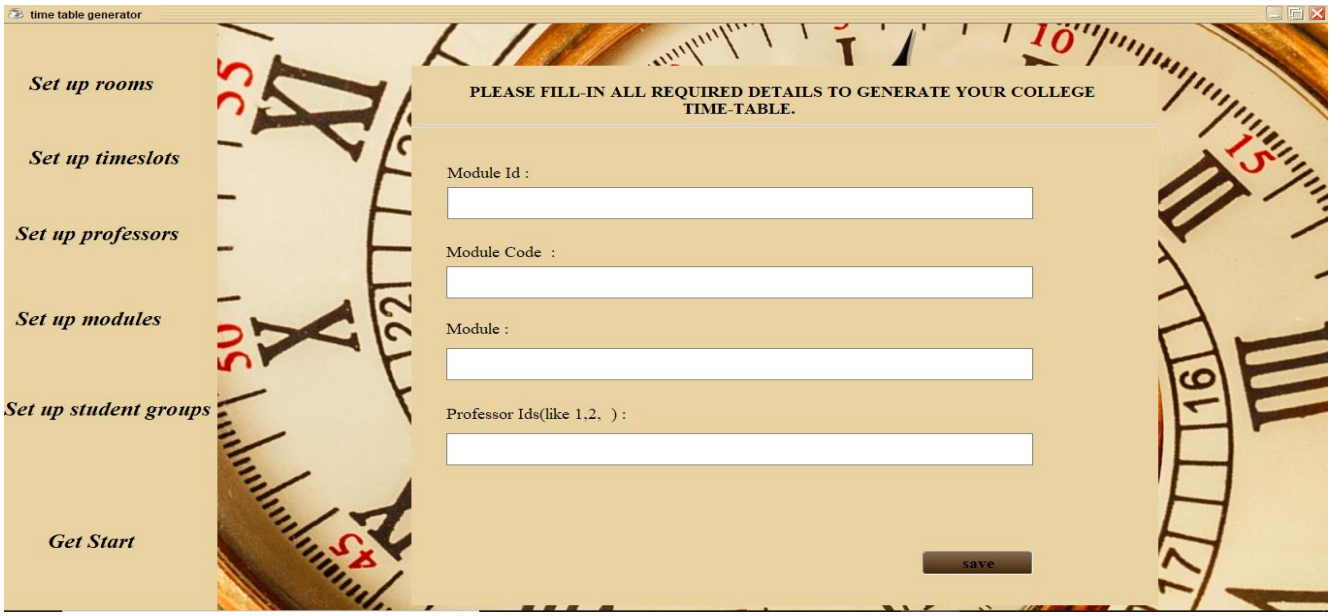



*Figure 29 Set up modules page*

Fill in the information and for each module

1. module Id
2. module code
3. module
4. professor IDs

Clicking the save Button when you have finished all the module go to set up student groups


*Figure 30 set up student groups*

Fill in the information and for each student groups

1. groups Id
2. groups size
3. module IDs

After entering the information, click the get start button to get the results

The information entered is saved in json file the result is also saved in json file


*Figure 31 output time table*

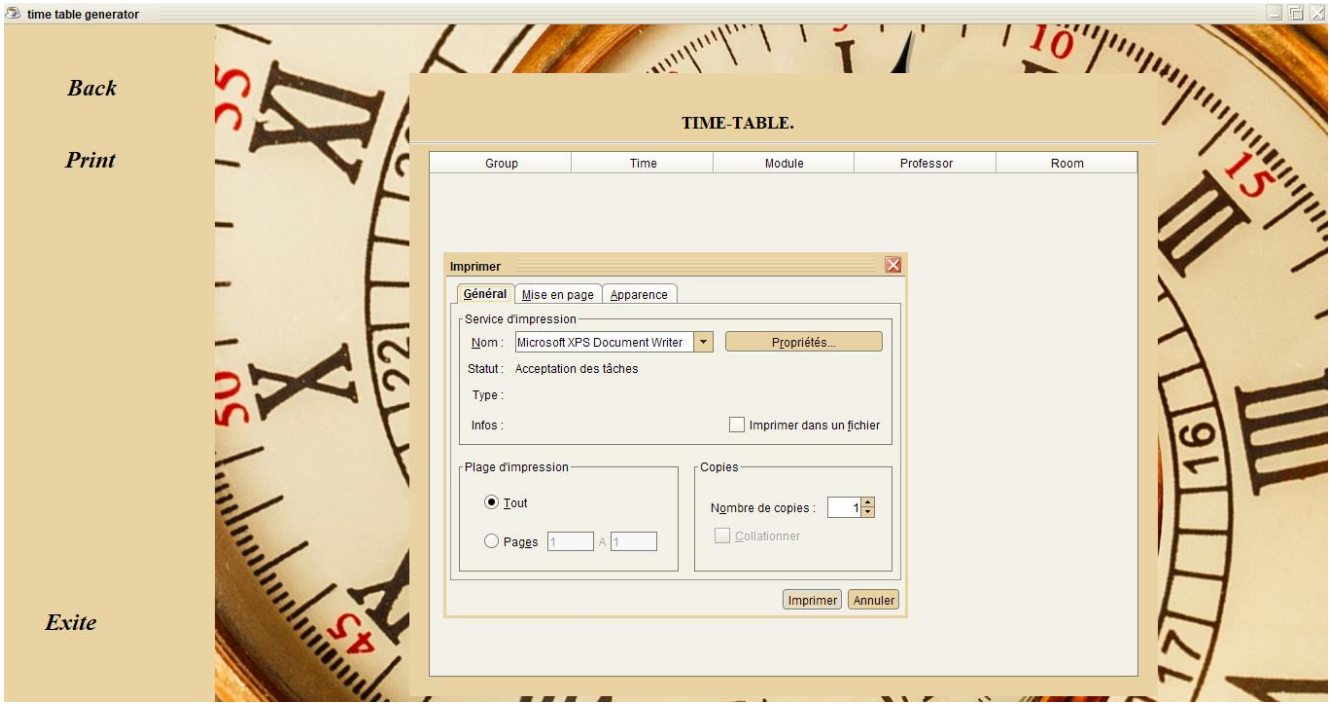



*Figure 32 Print the output time table*

Result can be printed Clicking the Print Button

# Test Data

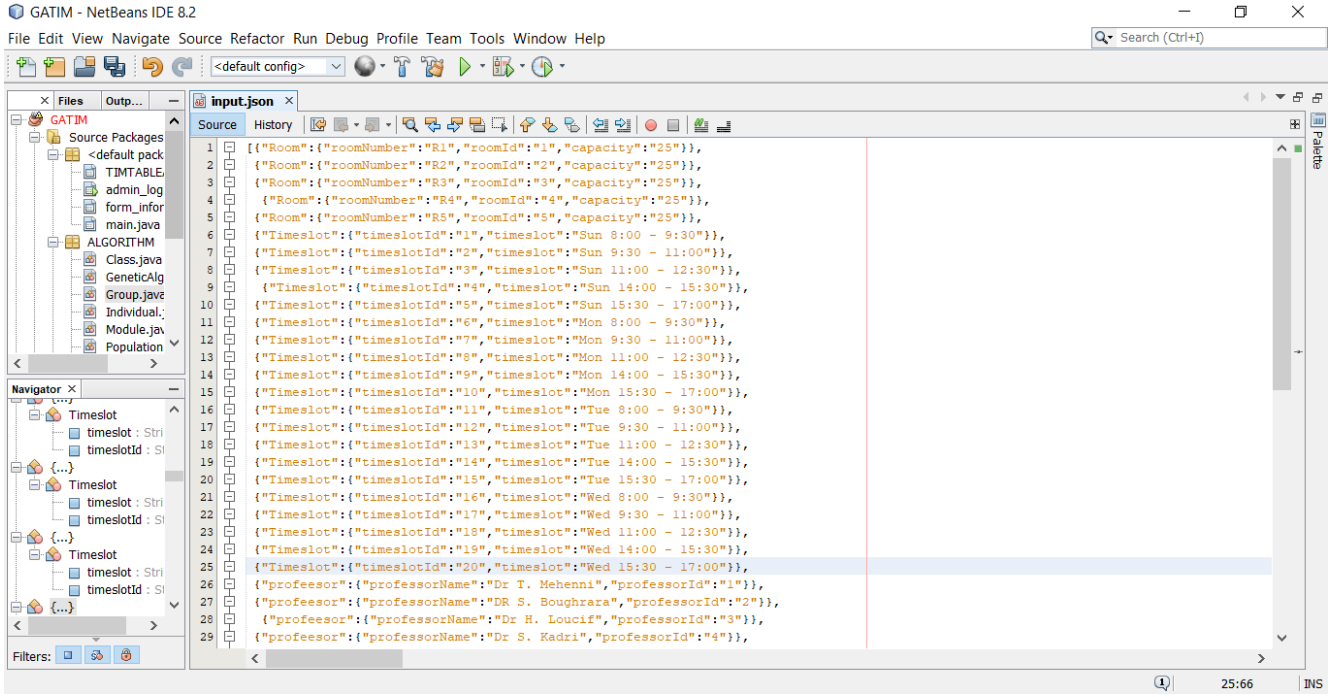



*Figure 33 Data test*

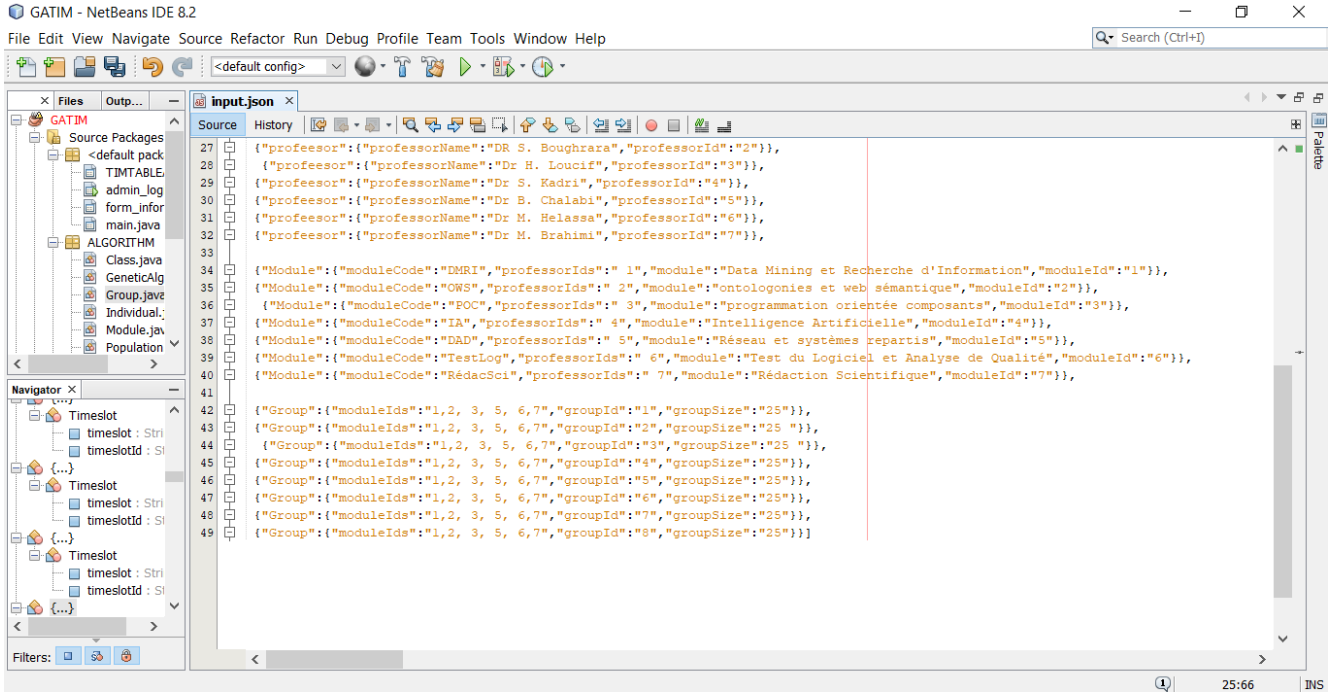
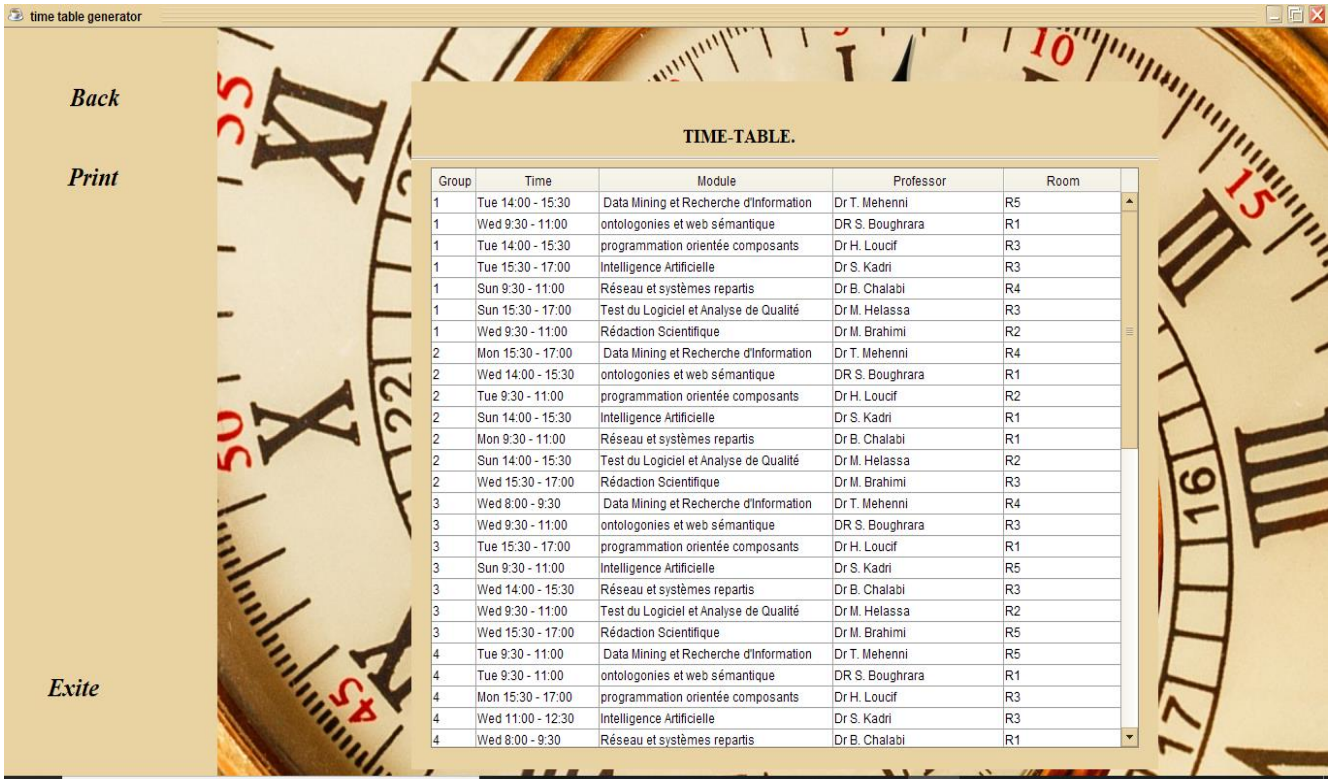


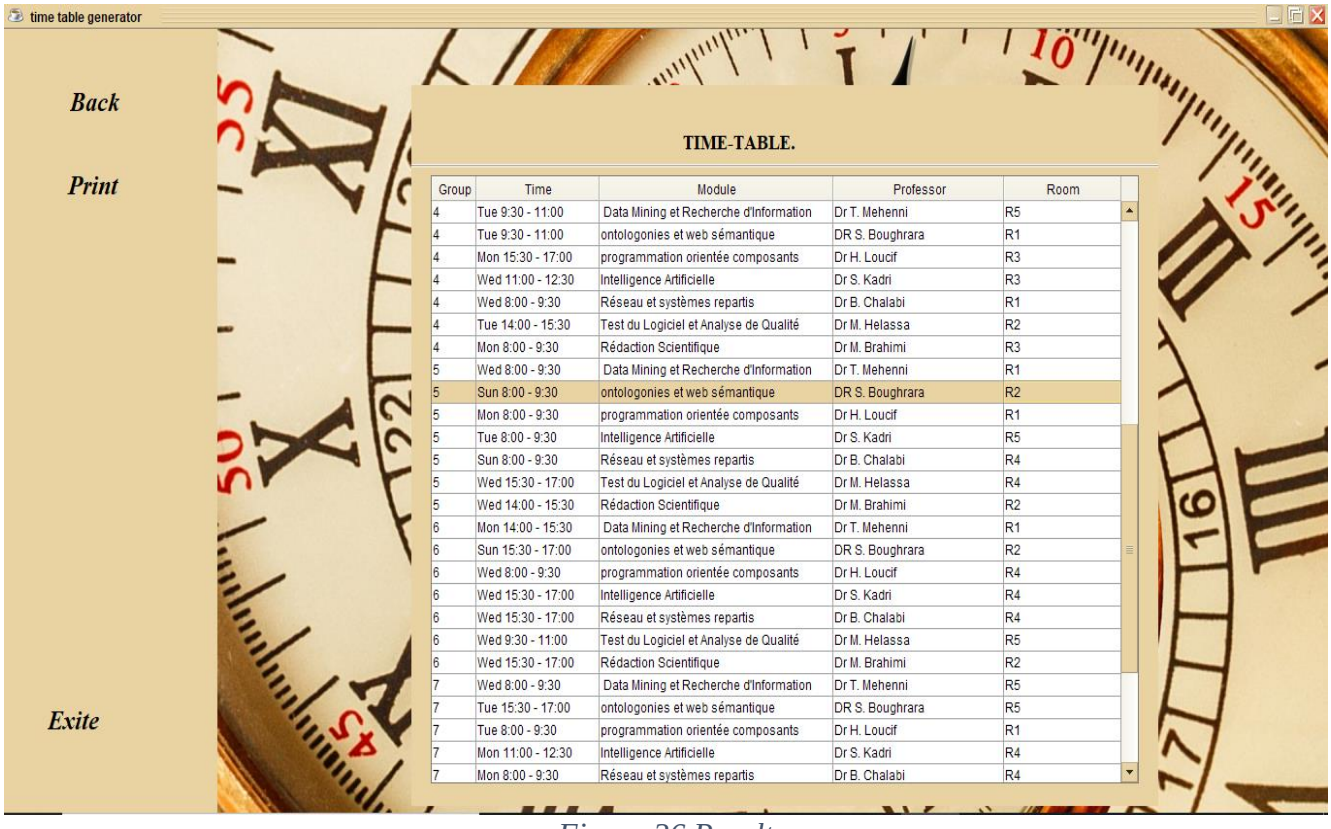

*Figure 34 Data test*

# Results



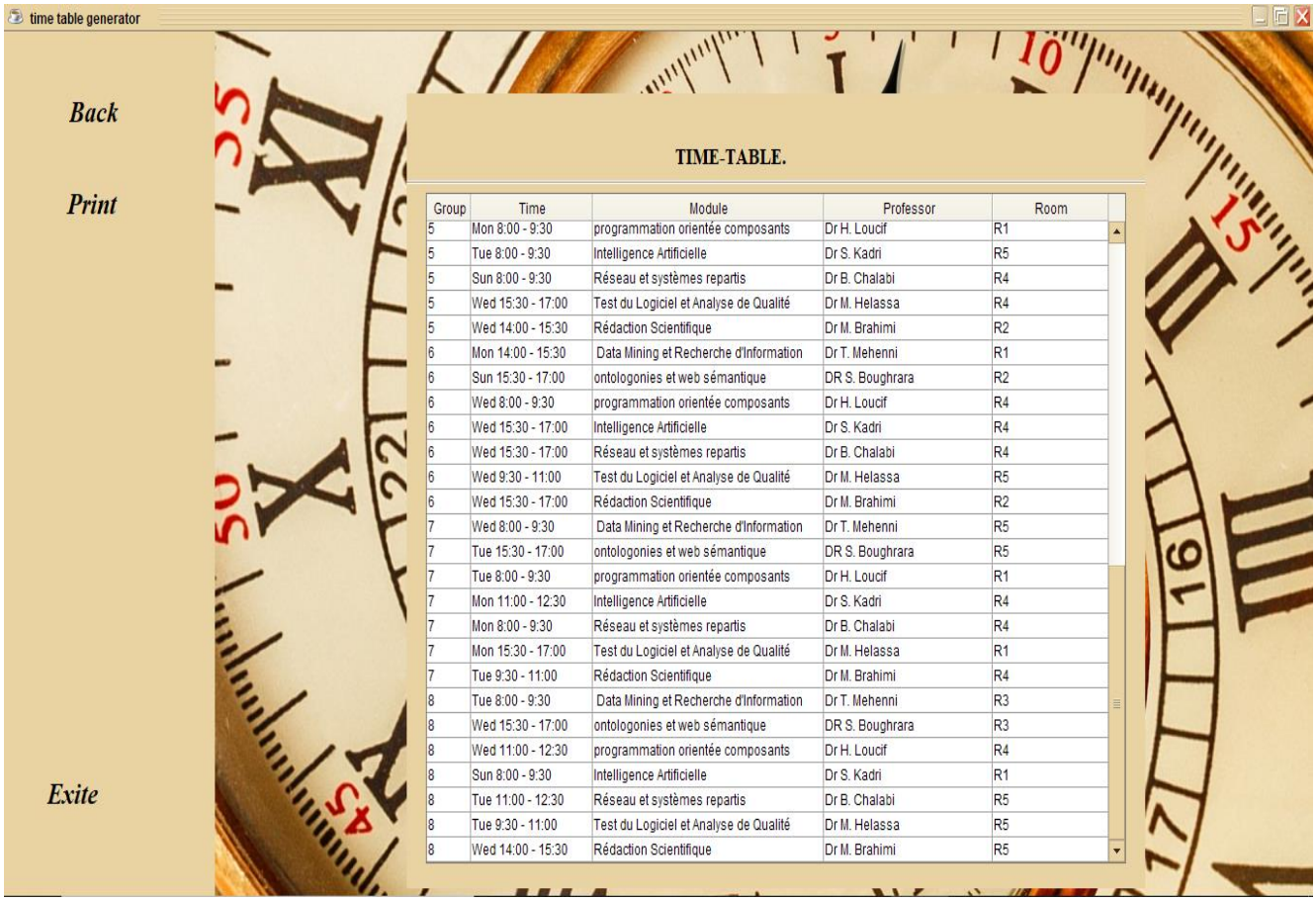

| Group | Time | Module | Professor | Room |
|---|---|---|---|---|
| 1 | Tue 14:00 - 15:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R5 |
| 1 | Wed 9:30 - 11:00 | ontologonies et web sémantique | DR S. Boughrara | R1 |
| 1 | Tue 14:00 - 15:30 | programmation orientée composants | Dr H. Loucif | R3 |
| 1 | Tue 15:30 - 17:00 | Intelligence Artificielle | Dr S. Kadri | R3 |
| 1 | Sun 9:30 - 11:00 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |
| 1 | Sun 15:30 - 17:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R3 |
| 1 | Wed 9:30 - 11:00 | Rédaction Scientifique | Dr M. Brahimi | R2 |
| 2 | Mon 15:30 - 17:00 | Data Mining et Recherche d'Information | Dr T. Mehenni | R4 |
| 2 | Wed 14:00 - 15:30 | ontologonies et web sémantique | DR S. Boughrara | R1 |
| 2 | Tue 9:30 - 11:00 | programmation orientée composants | Dr H. Loucif | R2 |
| 2 | Sun 14:00 - 15:30 | Intelligence Artificielle | Dr S. Kadri | R1 |
| 2 | Mon 9:30 - 11:00 | Réseau et systèmes repartis | Dr B. Chalabi | R1 |
| 2 | Sun 14:00 - 15:30 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R2 |
| 2 | Wed 15:30 - 17:00 | Rédaction Scientifique | Dr M. Brahimi | R3 |
| 3 | Wed 8:00 - 9:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R4 |
| 3 | Wed 9:30 - 11:00 | ontologonies et web sémantique | DR S. Boughrara | R3 |
| 3 | Tue 15:30 - 17:00 | programmation orientée composants | Dr H. Loucif | R1 |
| 3 | Sun 9:30 - 11:00 | Intelligence Artificielle | Dr S. Kadri | R5 |
| 3 | Wed 14:00 - 15:30 | Réseau et systèmes repartis | Dr B. Chalabi | R3 |
| 3 | Wed 9:30 - 11:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R2 |
| 3 | Wed 15:30 - 17:00 | Rédaction Scientifique | Dr M. Brahimi | R5 |
| 4 | Tue 9:30 - 11:00 | Data Mining et Recherche d'Information | Dr T. Mehenni | R5 |
| 4 | Tue 9:30 - 11:00 | ontologonies et web sémantique | DR S. Boughrara | R1 |
| 4 | Mon 15:30 - 17:00 | programmation orientée composants | Dr H. Loucif | R3 |
| 4 | Wed 11:00 - 12:30 | Intelligence Artificielle | Dr S. Kadri | R3 |
| 4 | Wed 8:00 - 9:30 | Réseau et systèmes repartis | Dr B. Chalabi | R1 |

*Figure 35 Result*




| Group | Time | Module | Professor | Room |
|---|---|---|---|---|
| 4 | Tue 9:30 - 11:00 | Data Mining et Recherche d'Information | Dr T. Mehenni | R5 |
| 4 | Tue 9:30 - 11:00 | ontologonies et web sémantique | DR S. Boughrara | R1 |
| 4 | Mon 15:30 - 17:00 | programmation orientée composants | Dr H. Loucif | R3 |
| 4 | Wed 11:00 - 12:30 | Intelligence Artificielle | Dr S. Kadri | R3 |
| 4 | Wed 8:00 - 9:30 | Réseau et systèmes repartis | Dr B. Chalabi | R1 |
| 4 | Tue 14:00 - 15:30 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R2 |
| 4 | Mon 8:00 - 9:30 | Rédaction Scientifique | Dr M. Brahimi | R3 |
| 5 | Wed 8:00 - 9:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R1 |
| 5 | Sun 8:00 - 9:30 | ontologonies et web sémantique | DR S. Boughrara | R2 |
| 5 | Mon 8:00 - 9:30 | programmation orientée composants | Dr H. Loucif | R1 |
| 5 | Tue 8:00 - 9:30 | Intelligence Artificielle | Dr S. Kadri | R5 |
| 5 | Sun 8:00 - 9:30 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |
| 5 | Wed 15:30 - 17:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R4 |
| 5 | Wed 14:00 - 15:30 | Rédaction Scientifique | Dr M. Brahimi | R2 |
| 6 | Mon 14:00 - 15:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R1 |
| 6 | Sun 15:30 - 17:00 | ontologonies et web sémantique | DR S. Boughrara | R2 |
| 6 | Wed 8:00 - 9:30 | programmation orientée composants | Dr H. Loucif | R4 |
| 6 | Wed 15:30 - 17:00 | Intelligence Artificielle | Dr S. Kadri | R4 |
| 6 | Wed 15:30 - 17:00 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |
| 6 | Wed 9:30 - 11:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R5 |
| 6 | Wed 15:30 - 17:00 | Rédaction Scientifique | Dr M. Brahimi | R2 |
| 7 | Wed 8:00 - 9:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R5 |
| 7 | Tue 15:30 - 17:00 | ontologonies et web sémantique | DR S. Boughrara | R5 |
| 7 | Tue 8:00 - 9:30 | programmation orientée composants | Dr H. Loucif | R1 |
| 7 | Mon 11:00 - 12:30 | Intelligence Artificielle | Dr S. Kadri | R4 |
| 7 | Mon 8:00 - 9:30 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |

*Figure 36 Result*




| Group | Time | Module | Professor | Room |
|---|---|---|---|---|
| 5 | Mon 8:00 - 9:30 | programmation orientée composants | Dr H. Loucif | R1 |
| 5 | Tue 8:00 - 9:30 | Intelligence Artificielle | Dr S. Kadri | R5 |
| 5 | Sun 8:00 - 9:30 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |
| 5 | Wed 15:30 - 17:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R4 |
| 5 | Wed 14:00 - 15:30 | Rédaction Scientifique | Dr M. Brahimi | R2 |
| 6 | Mon 14:00 - 15:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R1 |
| 6 | Sun 15:30 - 17:00 | ontologonies et web sémantique | DR S. Boughrara | R2 |
| 6 | Wed 8:00 - 9:30 | programmation orientée composants | Dr H. Loucif | R4 |
| 6 | Wed 15:30 - 17:00 | Intelligence Artificielle | Dr S. Kadri | R4 |
| 6 | Wed 15:30 - 17:00 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |
| 6 | Wed 9:30 - 11:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R5 |
| 6 | Wed 15:30 - 17:00 | Rédaction Scientifique | Dr M. Brahimi | R2 |
| 7 | Wed 8:00 - 9:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R5 |
| 7 | Tue 15:30 - 17:00 | ontologonies et web sémantique | DR S. Boughrara | R5 |
| 7 | Tue 8:00 - 9:30 | programmation orientée composants | Dr H. Loucif | R1 |
| 7 | Mon 11:00 - 12:30 | Intelligence Artificielle | Dr S. Kadri | R4 |
| 7 | Mon 8:00 - 9:30 | Réseau et systèmes repartis | Dr B. Chalabi | R4 |
| 7 | Mon 15:30 - 17:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R1 |
| 7 | Tue 9:30 - 11:00 | Rédaction Scientifique | Dr M. Brahimi | R4 |
| 8 | Tue 8:00 - 9:30 | Data Mining et Recherche d'Information | Dr T. Mehenni | R3 |
| 8 | Wed 15:30 - 17:00 | ontologonies et web sémantique | DR S. Boughrara | R3 |
| 8 | Wed 11:00 - 12:30 | programmation orientée composants | Dr H. Loucif | R4 |
| 8 | Sun 8:00 - 9:30 | Intelligence Artificielle | Dr S. Kadri | R1 |
| 8 | Tue 11:00 - 12:30 | Réseau et systèmes repartis | Dr B. Chalabi | R5 |
| 8 | Tue 9:30 - 11:00 | Test du Logiciel et Analyse de Qualité | Dr M. Helassa | R5 |
| 8 | Wed 14:00 - 15:30 | Rédaction Scientifique | Dr M. Brahimi | R5 |

*Figure 37 Result*

# Conclusion

At the end of the modeling and the realization, the result will be an application that first allows the generation of the schedules of the different promotions and teachers without any conflict at the level of the sessions. This application will have made it possible to meet the needs of the administrative staff by solving the problems of the management of schedules as well as the conflicts that may exist.

# General conclusion.

Creating a timetable at the level of schools as well as universities takes great effort and time, and sometimes lacks accuracy. Therefore, in our work, we touched on the completion of an application that depends mainly on genetic algorithms to complete a timetable, as it is the future of digitization, especially at the level of public administration.

It mainly depends on the collection of various soft and hard constraints that aim to maintain these conditions and come up with the best solution. First, information about professor (professor id , professor name ) Room ( RoomID, rom Nnumber,Capasity) Module (ModuleCode , moduleID ) and group (groupId , groupSIze ) is inserted, and based on what we have presented, the genetic algorithm works to infer the timing The appropriate time with the possibility of printing as well.

This type of application contributes to the modernization of the administrations, which saves them from effort and time, and gives the character of digitization due to the scientific competence that the university holds in various fields, so that these services are to and from the student, as it is desirable that the graduation topics include works that the university exploits by itself and gives That is the face and footprint of the university.

Solving the problem of preparing a timetable for the university, ending the era of papers and starting the era of digitization, especially at the level of our college, and it requires us to prepare the completion of a database for each student separately in order to follow up on his academic life (exam points, discipline and perseverance, sick holidays, achievements and congratulations ....), which ends the often tiring problem of archiving, which requires a precise system at work, and it also facilitates the work of the administration. Be a follow-up of the student and know everything around him by accessing his account only at the level of archiving at the university administration

# Bibliography


[1] B. salem, «Digitization in Algerian university libraries,» *Cybrarians journal*, vol. 21, pp. 1687 - 2215, 2019.

[2] D. Werra, "An introduction to timetabling," *European Journal of Operational Research*, pp. 151-162, 1985.

[3] A. Wren, "Practice and Theory of Automated Timetabling," in *In the Practice and Theory of Automated*, Springer- Verlag, 1996.

[4] B. E, K. J, . J. K et W. R, «Automated University Timetabling: The State of the Art,» *The Computer Journal*, vol. 40, n° %19, pp. 565-571, 1997.

[5] A. S et M. M, «University timetabling using,» *Journal of Applied Artificial*, 1999..

[6] R. Barták, «Dynamic Constraint Models for Planning and,» *In New Trends in Constraints*, pp. 237-255, 2000.

[7] W. Kocjan, «Dynamic scheduling,» RISE - Research Institutes of Sweden, ICT, SICS, Sweden, 2002..

[8] T. Müller, Constraint-based Timetabling, Prague, 2005.

[9] V. Mallawaarachi, «how to define a fitness function in a genetic algorithm',» *towards data science*, juin 2017.

[10] R. M. R. Lewis, Metaheuristics for University Course, 2006, pp. 01-03.

[11] S. Sivanandam et S. Deepa, «introduction to genetic algorithm,» p. 15.37, 2008.

[12] «neuraldesigner,» [En ligne]. Available: https://www.neuraldesigner.com/blog/genetic_algorithms_for_feature_selection.

[13] R. K.Ahuja, «Computers & Operations Research,» *sciencedirect*, vol. 27, n° %110, pp. 917-934, 2000.

[14] [En ligne]. Available: www.tutorialspoint.com.

[15] J. DY Cao, «Computer Technologyand development,» pp. 50--é, 2010.

[16] r. neevan, YTUPE CHANEL, 2020.

[17] D. E. .. Goldberg, «Genetic Algorithms, Tournament Selection,,» *Com plex Systems*, vol. 9, pp. 193- 212, 1995.

[18] Y.-F. Y. T.-C. S. Ming-Der Yang, « The Scientific World,» *Kai-Siang Huang The Scientific World Journal*, 2014.

[19] «International conference on Evolutionary Multi-Criterion Optimization,» chez *Multi-Criterion Optimization*, 2003.

[20] P. Y. Padmavathi kora, *international journal of computer aplications*, p. 162(10), 2017.

[21] D. K. .. m. hadush, «combined mutation operators of GA for the traveling salesman problem,» *international journal of combinatorial optimization problems and informatics*, 2007.

[22] K. J. J. H. K. R. W. E. Burke, «Automated University Timetabling: The State of the Art,» *The Computer Journal*, pp. 565-571, 1997.

[23] M. Carter, "*A Survey of Practical Applications of Examination Timetabling Algorithms*," *Operations Research*, vol. 34, pp. 193-202, 1986.

[24] G. White and W. Chan, "*Towards the Construction of Optimal Examination Schedules*," *INFOR*, vol. 17, pp. 219-229, 1979.

[25] M. Carter, G. Laporte, and S. Y. Lee, "*Examination Timetabling: Algorithmic Strategies and Applications*," *Journal of the Operational Research Society*, vol. 47, pp. 373-383, 1996.

[26] D. Brelaz, "*New methods to color the vertices of a graph*," *Commun. ACM*, vol. 22, pp. 251-256 1979.
[27] B. Deris, S. Omatu, H. Ohta, and D. Samat, "*University Timetabling by Constraintbased Reasoning: A Case Study*," *Journal of Operational Reasearch Society*, vol. 48,pp. 1178-1190, 1997.
[28] G. Lajos, "*Complete University Modular Timetabling using Constraint LogicProgramming*," in *Practice and Theory of Automated Timetabling (PATAT) I*, vol. 1153, *Lecture Notes in Computer Science*, E. Burke and P. Ross, Eds. Berlin: Springer-Verlag, 1996, pp. 146-161.
[29] P. Boizumault, Y. Delon, and L. Peridy, "*Logic Programming for Examination Timetabling*," *Logic Program.*, vol. 26, pp. 217-233, 1996.
[30] M. Carter, "*A Langarian Relaxation Approach to the Classroom Assignment Problem*," *INFOR*, vol. 27, pp. 230-246, 1986.
[31] A. Tripathy, "*School Timetabling - A Case in Large Binary Linear Integer Programming* " *Managment Science*, vol. 30, pp. 1473-1489, 1984.
[32] S. Daskalaki, T. Birbas, and E. Housos, "*An integer programming formulation for a case study in university timetabling*," *European Journal of Operational Research*, vol. 153, pp. 117-135, 2004.
[33] http://www.metaheuristics.org (Website of the Metaheuristics Network)

[34] J. Freeman, « InfoWorld AUG 8,» 8 AUG 2019 . [En ligne]. Available:

https://www.infoworld.com.

## Abstract

This work is interested in Design and Realization of a Java API for Time Management for the University. The objective of this application is to facilitate the development of timetables at the level of the departments, using genetic algorithms.



**Keywords:**

Time management, java API, genetic algorithms



**Résumé**

Ce travail s'intéresse à Conception Et Réalisation d'un API java pour Gestion des Emplois du Temps pour l'université. L'objectif de cette application est de faciliter, l'élaboration des emplois du temps au niveau des départements, utilisant des algorithmes génétiques.



**Mots clés :**

Gestion des Emplois du Temps, API java, algorithmes génétiques



**ملخص**

يهتم هذا العمل بتصميم وإنجاز واجهة برمجة تطبيقات Java لإدارة الوقت للجامعة. الهدف من هذا التطبيق هو تسهيل تطوير الجداول الزمنية على مستوى الجامعة، باستخدام الخوارزميات الجينية.



**الكلمات الدالة :**

إدارة الوقت ، جافا API ، الخوارزميات الجينية.