Marco Antonio Luna Calvillo

Instituto Tecnológico y de Estudios Superiores de Monterrey

Abstract

Vademecum that gives you prescription depending on your symptoms in a variety of countries using Prolog

International vademecum

Programming Languages



1. **Context of the Problem**

**a. Diseases do not discriminate**

Imagine the following scenario:

You go for trip to other country. You go to study, work, pleasure, etc. For weeks (even months) you have been planning your trip and saved money to do all kind of stuff. Must of the risks have been mitigated. Everything is ready.

When you arrive to your destination it’s everything you imagine. The city’s vibe is great, the activities planned are as great as you imagine, no, even better. You are having a really good time; all your efforts are worth it.

You wake up next morning and you realized not everything is rosy. You are not feeling as well as last night. You feel a little bit dizzy, you have stomach ache (the last rounds at the bar or that street hot dog may not be the best idea).

Maybe that’s not you, let’s get to another scenario. You wake up and you have sore throat and fever, all because you thought that sweater wasn’t going to be necessary. You wake up next morning, you are not as healthy as last night.

You are a smart person, you get international insurance if anything happened. When you checked it, it covers mostly everything, even death, but not a silly fever or stomach ache. You feel sick but not as sick to go to hospital.

So, you go to the drug store to buy some medicine. The language is familiar to you since you were little (assuming you took classes). Feeling like zombie, you arrive to the drug store and start looking for your favorite Mexican medicine, that one your mother has gave you your entire life.

Here comes the best part, they don’t have it! What did you expect? Not all medicines are sold around the world and you don’t recall the medicine active ingredient. So, you start talking to the guy in the drugstore hoping he gives you the solution you are looking to continue enjoying your trip.

I lived this scenario many times (more that I want to admit) and I start carrying my med kit everywhere I go. But (there’s always a but), for some reason, that wasn’t enough. I ended in a drugstore asking the employees for Tempra…

I didn’t need to upgrade my med kit hoping I survive my next trip, I need a solution that covers every situation. Here’s is where my solution was born.

**b. The complexity of a DB Query**

One of the most important part of software development is the architecture, more specific, the database.

The database is going to define your project. I have been developing for about three years and I realized that the database is one of the most critical elements of the solution. Is the core of your system, the *brain* and what your solution is capable to do.

Changing the database is one of the most complex things to do. Carlo Batini explained in his book *Conceptual Database Design: An Entity-Relationship Approach* that trying to change a database is a delicate task and must be done carefully.

*“(…) database design plays a central role in the information resource management of most organizations”* (Batini, C. 1992). This quote explains how delicate the database is. A great database allows you to manipulate the data in a very efficient way. The point of a good database is for you to play with the information to analyze the past, the present and predict the future.

Once you have the data, you must play with it using queries. Queries allows to read, transform, join or edit the data contained in the database. The database is the bank and the queries are the keys to access the vault.

When you have complex database, you need to have complex queries to access them. The biggest database I have work with had around 1,200 tables and being completely honest, I made a poor use of it. I had to do complex queries to get the data doing a lot of joins and most of them crash or didn’t work as expected because I didn’t have the knowledge to do them.

As I tried for the 10th time to make that query work, I thought there needs to be a better and easier way to make complex queries. This is when I found another way to storage information.

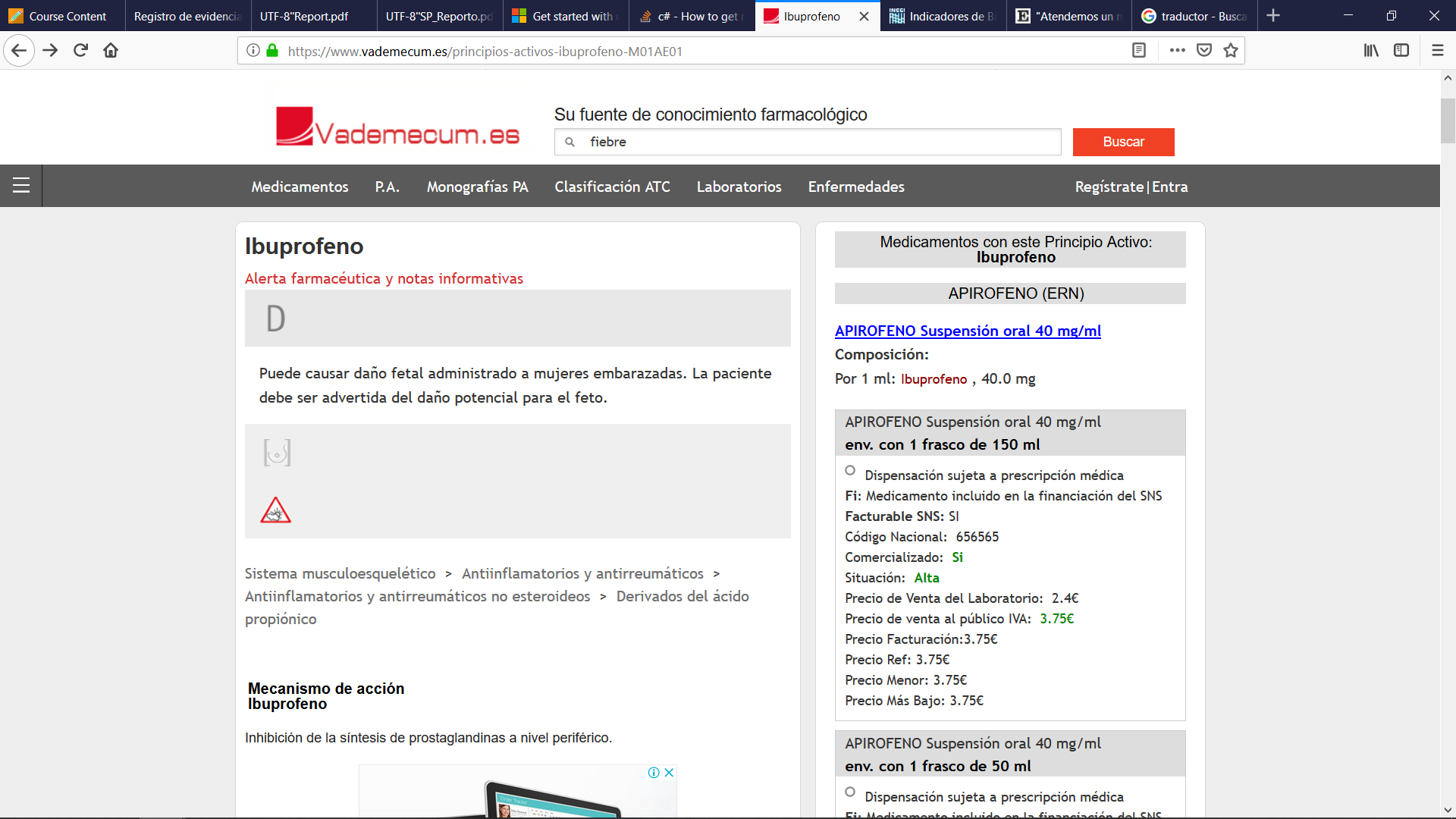
1. **Solution and Results**

My project consists in an International Vademecum using Prolog as Database.

A *Vademecum* is a compendium of presentations, compositions y principal indications of medicines. Every doctor has one of it that they use as reference and it is *not legal* for a common person to acquire it.

There are a lot of Vademecum websites to auto-medicate yourself. The problem with them is that are very difficult to understand because of the detail level it had.

In *IMG. 1*, you can see a little bit of the information about one of the most common medications there are: ibuprofen. There is a lot of information that may not be necessary for a regular person; most of it is scientific information for doctors.

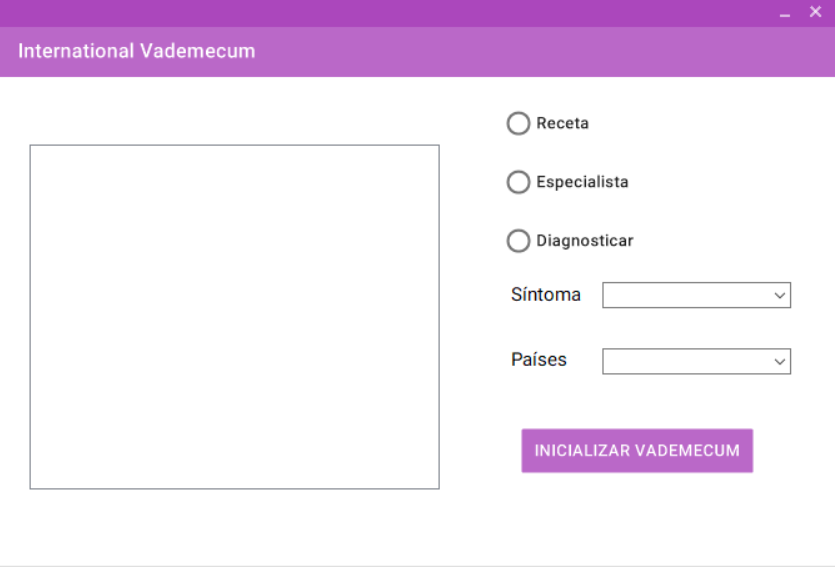


IMG. 1 Vademecum.es - Ibuprofen

Trying to read and to find useful information is kind of difficult because of the detail level. You can spend a lot time trying to determinate which is the best medicine for you and you could also end with a very potent prescription and end worst.

My solution has three options:

* **Receta:** It gives you a prescription depending on your symptoms and the country where you are.
* **Especialista:** It gives you who is the specialist or what doctor can treat you depending on the symptom.
* **Diagnostico:** You can select a symptom or as much as three and, if exists, it gives you the disease that have all symptoms.



IMG. 2 International Vademecum

Before you start making all your consults, you must initialize it. You must check if all requirements are met (check the *setup instructions* section).

All the information is based on personal experience, family and friends. **All of this are recommendations**, any of this have a doctor validation but could get you out of an actual hurry in another country. The Regulations of the General Law of Health in Matters of Provision of Medical Care Services established that only licensed doctor can give diagnostics and make valid prescriptions. I repeat, the objective of this project is to give recommendations, not anything more.

The project use Prolog as the Database Engine. What I found out is that I saved a lot time doing the logic in Prolog rather than using SQL.

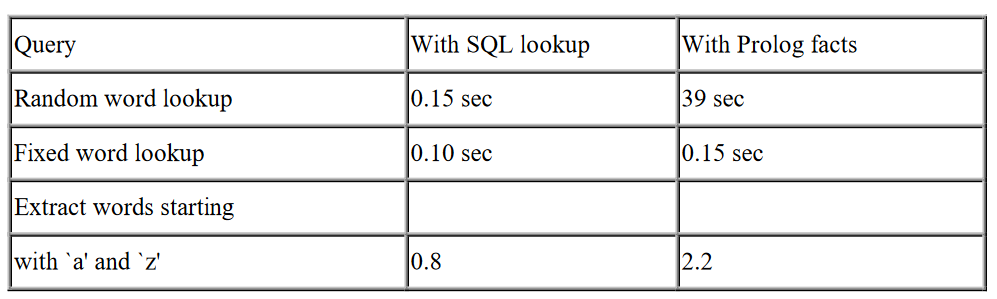
In Prolog, I had to declare a couple of rules to make complex queries. The facts are simples and the rules doesn’t need as much logic as SQL queries or tables. Here’s an example of a rule used:

*receta(Medicina,Sintoma,Pais):-sintomas(Sintoma,Z), medicina(Medicina,Z,Pais).*

Prolog inference is used to derive conclusions. The inference does backward chaining, it has a goal and several sub-goals. What it does is, for all the goals, Prolog proves or disproves each one of them.

Using SQL is a different story. The first thing that pops into my mind is the use of Primary Keys, Indexes for the symptoms and relationship between tables to have a structured design. The queries would be very complex to develop, you would need to iterate many times to get the diagnostic.

In a study made in Middlesex University, they compare the time spend looking for words in a Database using Prolog and SQL. The database was about 100,000 records of words. They only look for words, that start with some specific letter or just a random word. The results were the following:



In the following table we can see that SQL have a better performance with **simply queries**. The queries were only looking for words in one table. In Prolog, there where same amount of facts like these ones (not complexity, just quantity):

word(autumn).

word(horse).

1. **Conclusions**

This is the first time I made a project in Prolog. It was an interesting experience working with another engine analyzing the results in all the process. The analysis, design and the develop of the solution was very different from my other projects. In the following table you could see a comparison between Prolog and SQL:

|  |  |  |
| --- | --- | --- |
|  | Prolog | SQL |
| Advantages | * Complex Searches * Inference Engine * Fact Manipulation * Unification and Backtracking * Client Language | * Quick Searches * Relational Engine * Data Storage * Server Language |
| Disadvantages | * Needs to recompile with changes * Little compatibility with new technologies | * Not very efficient with Data Analysis (e.g. Fibonacci Series) * Hard Hardware dependence |

1. **Setup Instructions**

The project can be found in this [Github repository](https://github.com/lunamrcc/international_vademecum).

To execute the project, you need to have the following:

* SWI Prolog 6.6.1 (32 bits) [Included in the Project folder]

It must be that Prolog Version and must be the one of 32 bits. The reason is because of the Prolog Extension used in the project, SwiPlCs.

SwiPlCs is an engine that allows to execute Prolog with .NET framework. The last version was released in 2014 (1.1.60605.0) and was last tested with SWI-Prolog 6.6.5. (32 bits). In the documentation it is record that can be uses with the 64-bit version (not very stable), but it was used with .NET 2010. In my case, I try using the latest version of Prolog (7.6.4) and using Visual Studio 2018 but I have a lot of trouble.

The problem I had is that I wasn’t able to access the Prolog executable. It seems that in these past years, the way the memory is access has been changed because I had problems with the memory allocation, even with the 32-bit version.

The way to solve my problem is to use the versions that were one of the last versions that were tested with SwiPlCs.

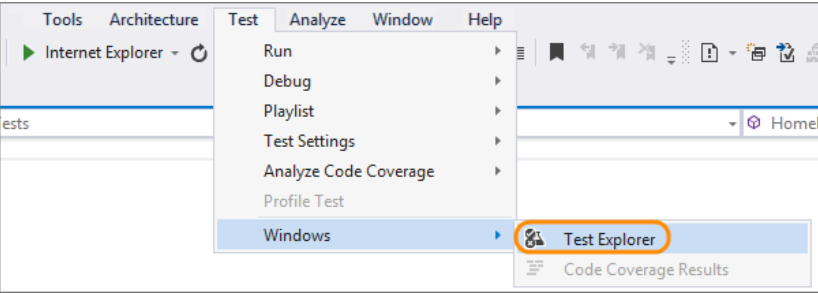
Once you have installed the necessary Prolog Version, copy the folder to C: and run the file *Vademecum.exe* in the main folder. First, you need to run tests to check that you have all components and then make the queries you want.

1. **Tests**

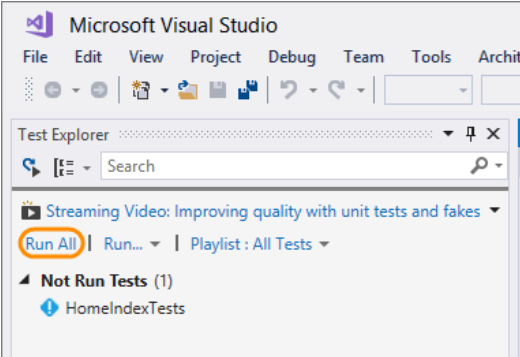
The solution contains a Test Project that test the connectivity with Prolog and some aspects of the Vademecum. To run the tests, you need to have Visual Studio installed.

To run the test you do the following:

* Open the project in Visual Studio.
* Select Test > Windows > Text Explorer



* Run Unit Tests



|  |  |
| --- | --- |
| Test Name | Description |
| Test\_Prolog\_x86\_Installed | Test that Prolog 32-bits is installed. |
| Test\_Prolog\_x86\_Version | Test that the version of Prolog 32-bits is 6.6.1. |
| Test\_Prolog\_x86\_DLL | Test that *libswipl.dll* (the dll that executes Prolog) is in the installation folder. |
| Test\_Vademecum\_Folder | Test that the Vademecum folder is in C: |
| Test\_Prolog\_Medicamentos\_Prolog\_File | Test that the file “Medicamentos.pl” exists in the Vademecum folder. |
| Test\_Prolog\_Medicamentos\_Database\_File | Test that the file “Medicamentos.bd” exists in the Vademecum folder. |
| Test\_Medicamentospl\_cargar | Test that “Medicamentos.bd” contains inside the rule *cargar.* |
| Test\_Prolog\_Initialize | Test Prolog library (SwiPlCs) |
| Test\_CountryComboBox\_Not\_Null | Test that the CountryComboBox have information |
| Test\_SyntomsComboBox\_Not\_Null | Test that the SymptomsComboBox have information |

1. **References**

Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual database design: an Entity-relationship approach* (Vol. 116). Redwood City, CA: Benjamin/Cummings.

Barbas, P. M. (2014). *U.S. Patent No. 8,635,206*. Washington, DC: U.S. Patent and Trademark Office.

Ibuprofeno. (2018, 5 January). Recovered November 12, 2018, from <https://www.vademecum.es/principios-activos-ibuprofeno-M01AE01>

Letsa, U. (2014, 28 May). SwiPlCs [Publication in Github]. Recovered November 1, 2018, from <https://github.com/SWI-Prolog/contrib-swiplcs>

Boone, K. *Using SQL with Prolog to improve performance with large databases*. Middlesex University, London. February 1999.