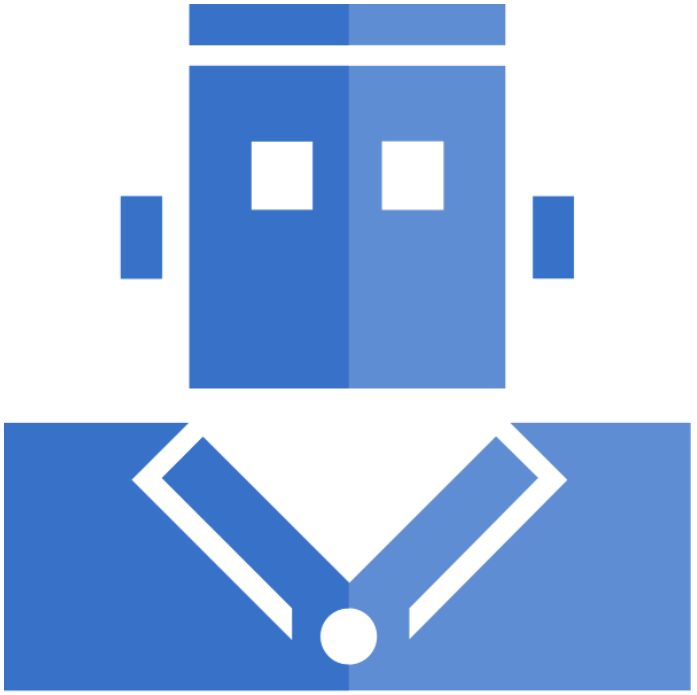
Institut Puig Castellar
Santa Coloma de Gramenet



PUIGBOT

Participants: Amanda Martínez
Sulman Ali
Sandra Navarro

2n CFGS DAM
Data lliurament: 01 - Juny - 2018

Resum del projecte:

Desenvolupament d'un bot de consulta d'informació per als estudiants i els pares de l'Institut Puig Castellar a través de l'aplicació Telegram. Instal·lació i configuració d'un Moodle 3.4 en una màquina virtual Ubuntu.

El bot haurà de tenir opcions per veure les notes, exàmens, entregues, i altre documentació d'interès per a l'usuari. El bot utilitzarà comandes simples com '/entregues' per poder accedir a la informació. Un alumne només podrà accedir a la seva pròpia informació, a través del seu usuari i contrasenya de Moodle. De la mateixa manera, els pares de l'alumne només podran accedir a la informació dels seus fills.

Abstract:

Development of an information bot for students and parents of the Puig Castellar Institute using the Telegram application. Installation and configuration of Moodle 3.4 on an Ubuntu virtual machine.

The bot must have options to see the marks, tests, activities and other documentation of interest of the student. The bot will use simple commands such as '/activities' to access the information. A student will only be able to access his own information. Just as the father of a student can only access the information of their child.

Paraules clau:

Bot, Telegram, Moodle, consultes, alumnes.

Índex

1. Introducció

1.1 Context i justificació del treball

Aquest projecte té com a finalitat la creació i programació en Java d'un bot amb accés a la base de dades del Moodle de l'Institut. D'aquesta manera, els alumnes, pares o professors poden consultar des del mòbil els horaris, les faltes d'assistència, notes, treballs pendents i altres opcions d'interès. El bot utilitza sentències simples com '/entregues' per accedir a les dades a través del xat de l'aplicació.

Els alumnes només tenen accés a les seves dades, així com els pares només poden accedir a la informació dels seus fills. D'aquesta manera es pretén agilitzar la comunicació entre els pares i els tutors.

Amb l'ús del bot, l'accés a les dades del Moodle serà més senzill i ràpid per als usuaris ja que no cal accedir a la web sinó que es poden consultar totes les dades des del mateix xat de Telegram.

1.2 Objectius del Treball

Els objectius principals del projecte són:

- Instal·lació del Moodle a una màquina virtual.
- Introducció de dades al Moodle.
- Creació del bot.
- Programació del bot.
- Proves amb el bot sense informació del Moodle.
- Connectar el bot amb el Moodle.
- Proves amb les comandes del bot per accedir a la informació del Moodle.
- Creació del manual per a l'usuari.
- Entrega de la màquina virtual, el codi font del bot, la memòria del projecte i la presentació.

1.3 Enfocament i mètode seguit

La nostra estratègia es desenvolupar una aplicació des de zero. D'aquesta manera es poden aplicar tots els conceptes apresos durant els dos anys de curs. Des de les bases de dades fins la creació d'un bot d'una aplicació mòbil, que funcioni adequadament; passant per la instal·lació de sistemes operatius i programari divers.

El mètode de desenvolupament a seguir és Waterfall, que passa per les diverses fases de desenvolupament: Anàlisi i planificació, disseny, implementació i proves finals, de manera que complirem els objectius en l'ordre especificat en la fase de planificació. També aplicarem tàctiques d'SCRUM i altres metodologies Agile apreses al curs.

1.4 Planificació del projecte

Planificació

Activitat	Descripció
Selecció del projecte	Selecció final del projecte i del grup.
Definició del projecte	Breu descripció del projecte i les tecnologies que utilitzarem.
Presa de requeriments	Obtenció dels arxius necessaris per a les fases de Disseny i Implementació.
Planificació temporal	Planificació del temps necessari per a cada estat del projecte.

Disseny

Activitat	Descripció
Disseny del diagrama de Gantt	Disseny inicial de la planificació temporal del projecte.
Disseny aplicació Java	Disseny de les classes del projecte amb els elements que corresponen a cada una d'elles.
Disseny dels diagrames de casos d'ús	Disseny dels diagrames de casos d'ús que necessitarem per planificar les accions del bot.
Disseny dels diagrames de classes	Disseny dels diagrames de classes de les comandes del bot.

Implementació

Activitat	Descripció
Creació Moodle	Instal·lació del Moodle a una màquina virtual.
Carregar dades al Moodle	Introducció de les dades necessàries d'alumnes i cursos al Moodle.
Desenvolupament aplicació Java	Desenvolupament de l'aplicació 'bot' amb Java.

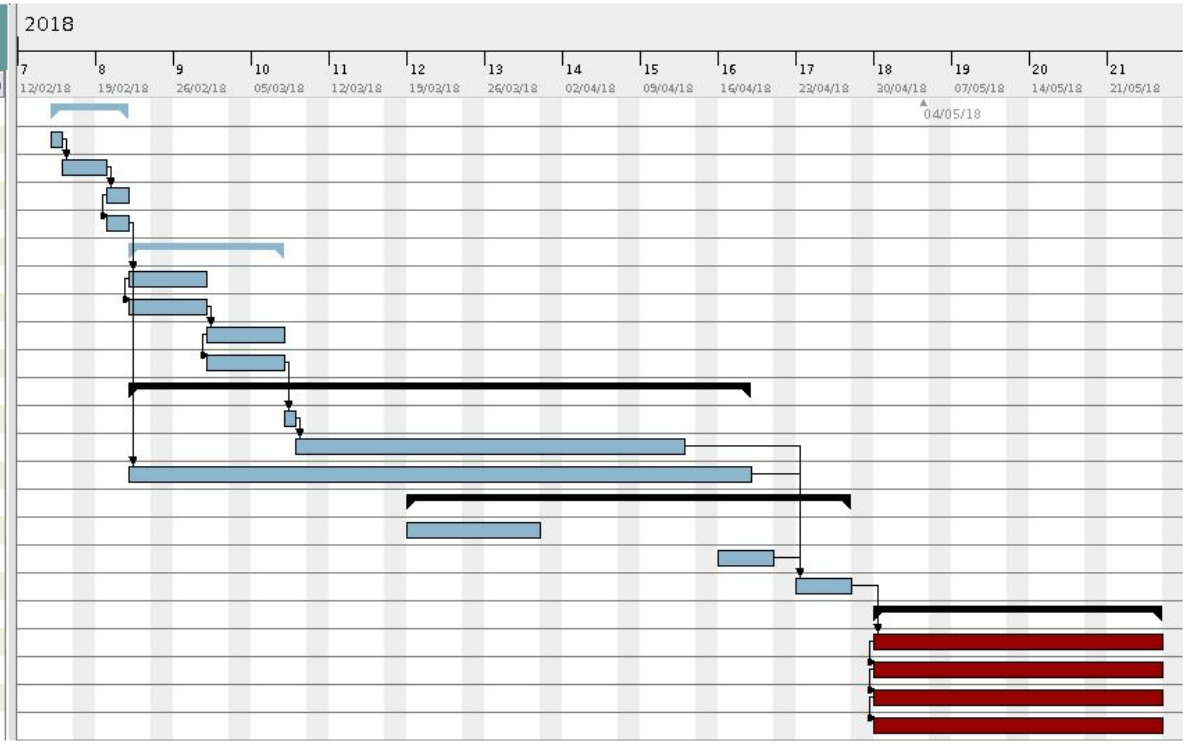
Proves

Activitat	Descripció
Proves inicials	Proves de l'aplicació per veure que les comandes funcionen adequadament.
Proves de connexió	Proves de l'aplicació per veure que es connecta adequadament amb el Moodle i agafa la informació necessària.
Prova final	Prova final de l'aplicació per veure que tot funciona adequadament.

Finalització

Activitat	Descripció
Redacció definitiva de la memòria	Redacció definitiva de la memòria
Preparació fitxers necessaris	Preparació dels fitxers necessaris per al funcionament adequat de l'aplicació.
Presentació visual	Creació de la presentació visual amb PowerPoint.
Revisió final	Revisió final de la memòria per comprovar que tot està ben finalitzat.

Nom	Inici	Finalització
Planificació	15/02/18	21/02/18
Selecció del projecte	15/02/18	15/02/18
Definició del projecte	16/02/18	19/02/18
Presa de requeriments	20/02/18	21/02/18
Planificació temporal	20/02/18	21/02/18
Disseny	22/02/18	07/03/18
Disseny diagrama de Gantt	22/02/18	28/02/18
Disseny aplicació Java	22/02/18	28/02/18
Disseny casos d'ús	01/03/18	07/03/18
Disseny diagrames de classes	01/03/18	07/03/18
Implementació	22/02/18	18/04/18
Creació Moodle	08/03/18	08/03/18
Carregar dades al Moodle	09/03/18	12/04/18
Desenvolupament aplicació Java	22/02/18	18/04/18
Proves	19/03/18	27/04/18
Proves inicials	19/03/18	30/03/18
Proves de connexió	16/04/18	20/04/18
Prova final	23/04/18	27/04/18
Finalització	30/04/18	25/05/18
Redacció definitiva memòria	30/04/18	25/05/18
Preparació fitxers necessaris	30/04/18	25/05/18
Presentació visual	30/04/18	25/05/18
Revisió final	30/04/18	25/05/18



1.5 Breu sumari de productes obtinguts

El projecte consta de:

- Una màquina virtual amb el Moodle.
- Un servidor amb el bot instal·lat i en funcionament.
- El codi font en Java del bot.
- El manual d'usuari del bot.
- La memòria del projecte amb tota la informació del procés de desenvolupament.
- La presentació per a la exposició davant del Tribunal que consta d'un resum del projecte, destacant els punts més importants.

1.6 Breu descripció dels altres capítols de la memòria

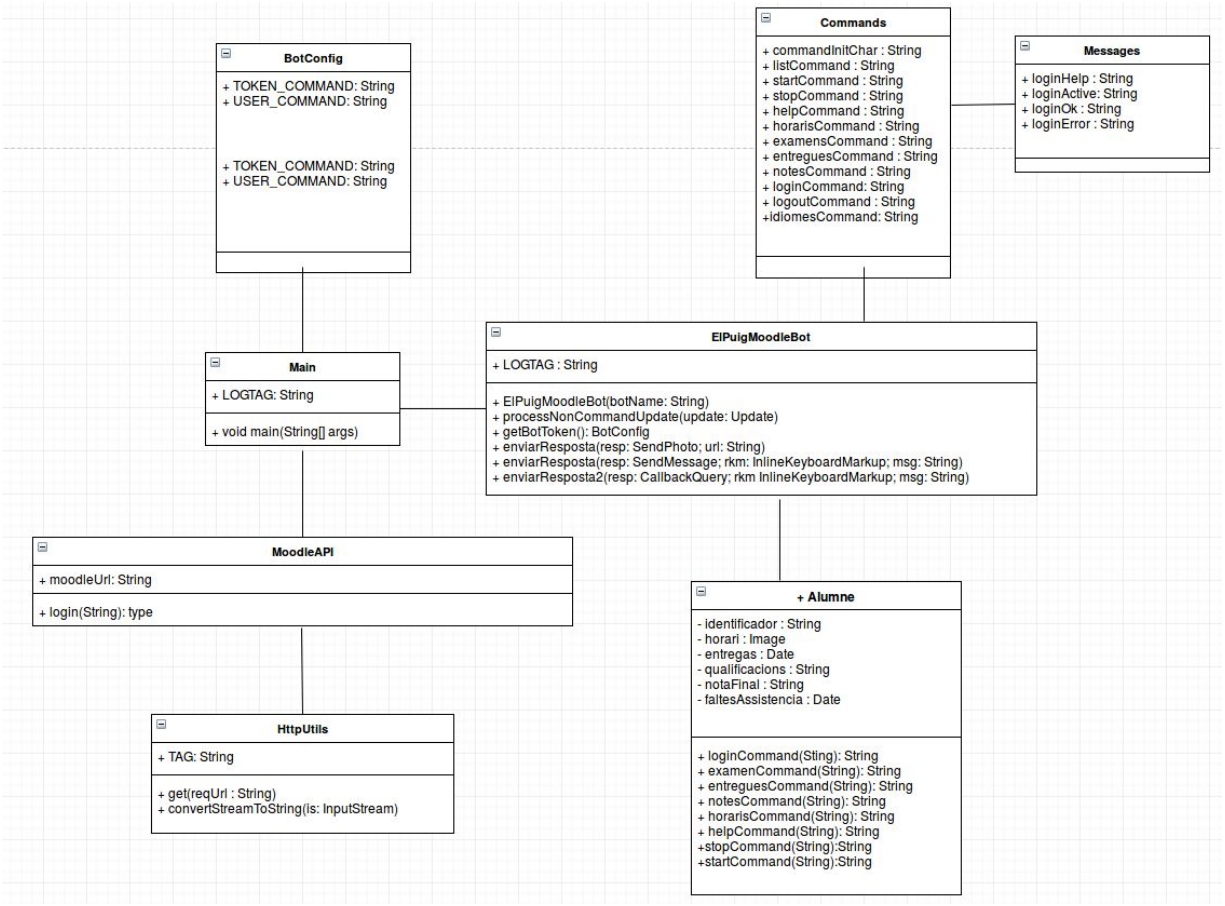
En els següents capítols s'explica el procés de desenvolupament del bot: les diverses fases de creació tant de la màquina virtual del Moodle com a del propi bot, i la connexió d'aquests dos. També s'inclou el manual d'usuari per a la utilització del bot i els dissenys.

1.7 Tecnologia i recursos utilitzats

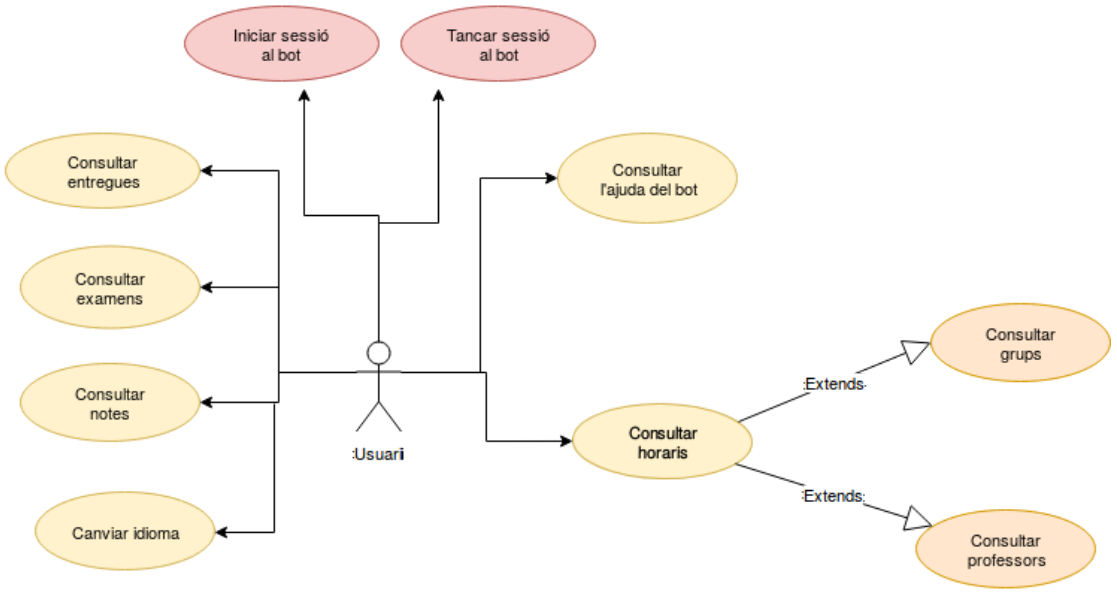
- Telegram
- Moodle 3.4
- Java
- Virtual Box
- Github
- Gant Project
- Google Drive
- Documentació Telegram
- Documentació Moodle
- IntelliJ

2. Dissenys

2.1 Disseny de l'aplicació Java

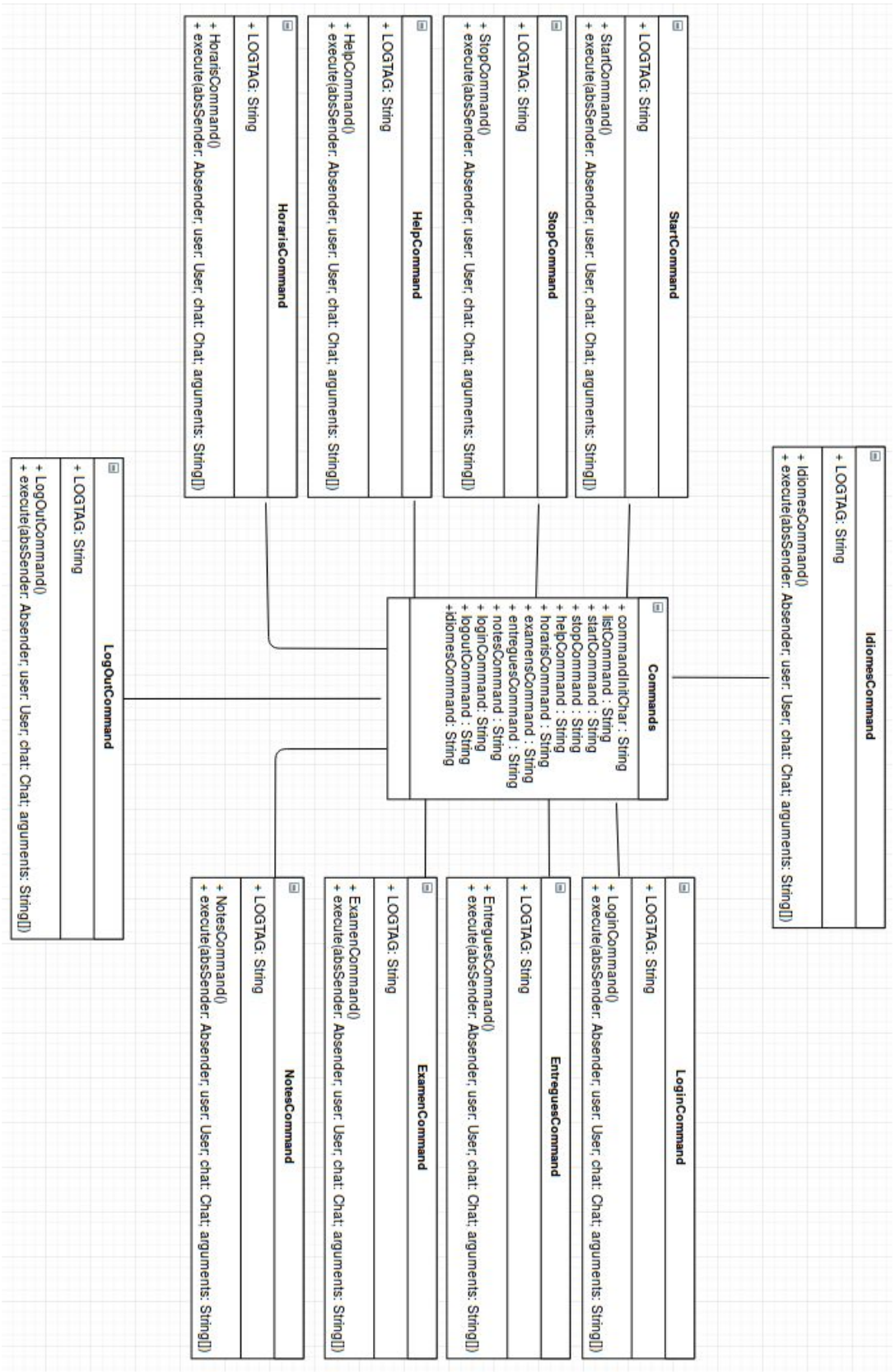


2.2 Disseny dels diagrames de casos d'ús

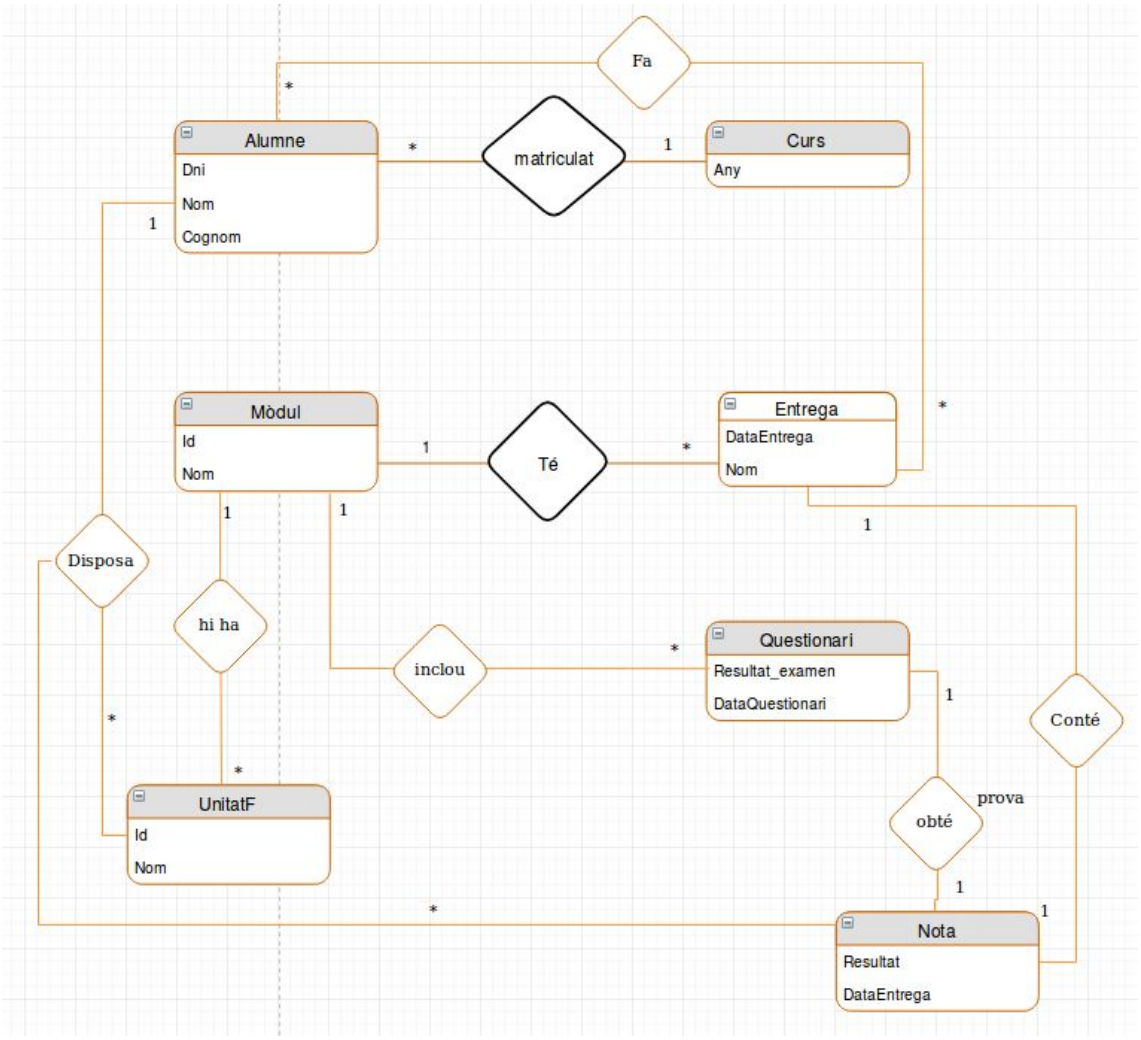


NOTA: Per poder accedir a totes les funcionalitats primer cal iniciar sessió.

2.3 Disseny dels diagrames de classes

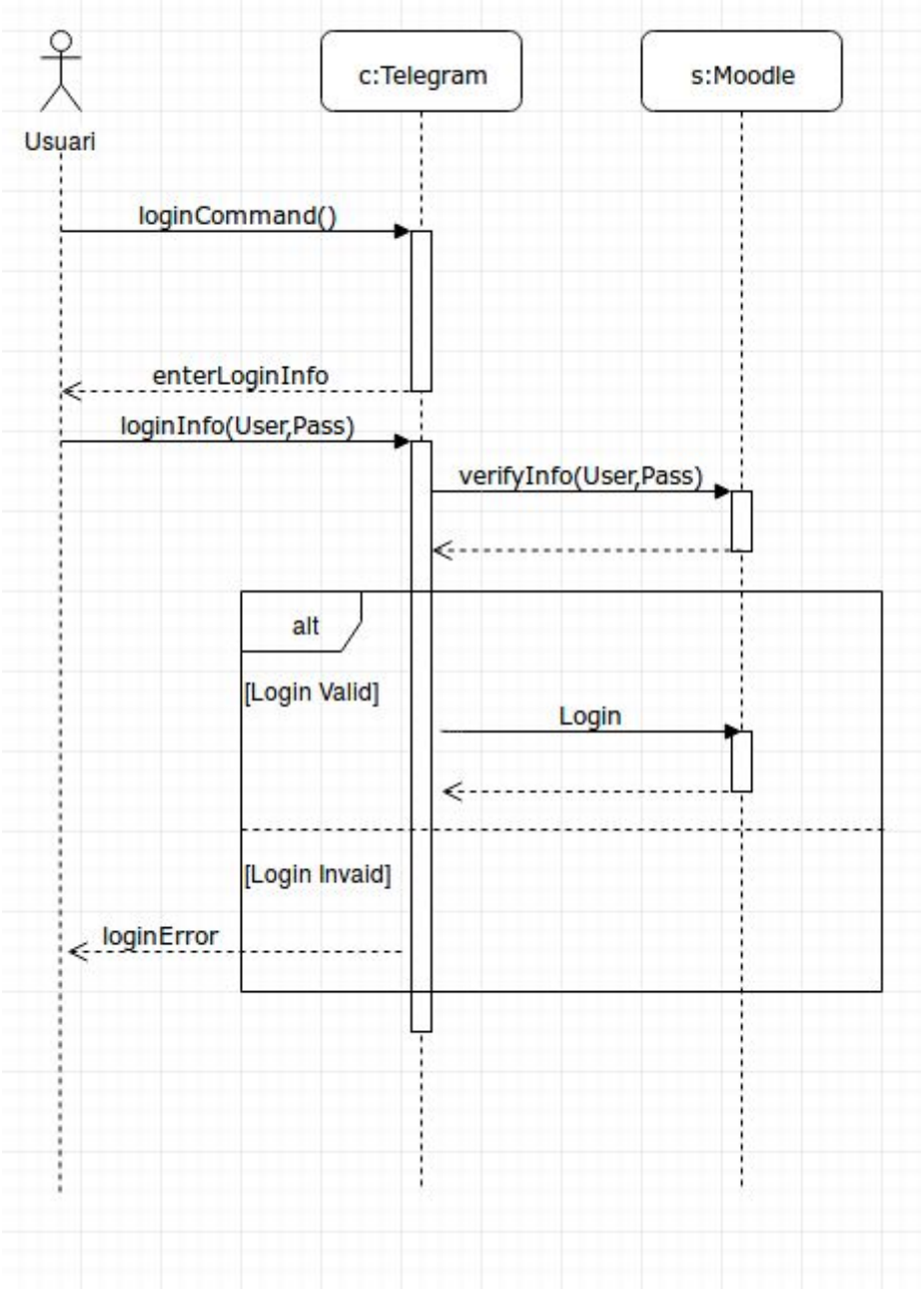


2.4 Disseny del diagrama entitat-relació

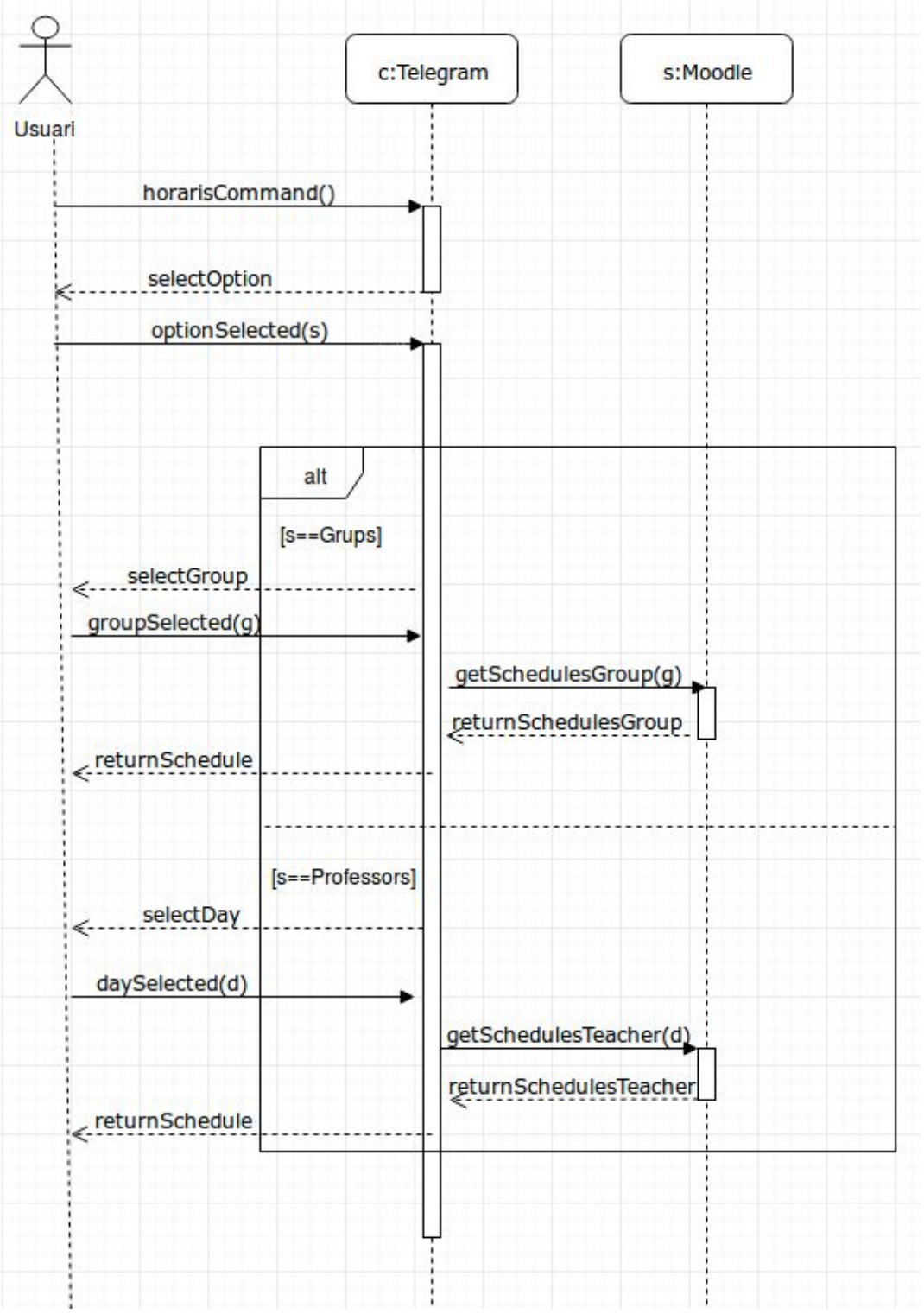


2.5 Disseny dels diagrames de seqüència

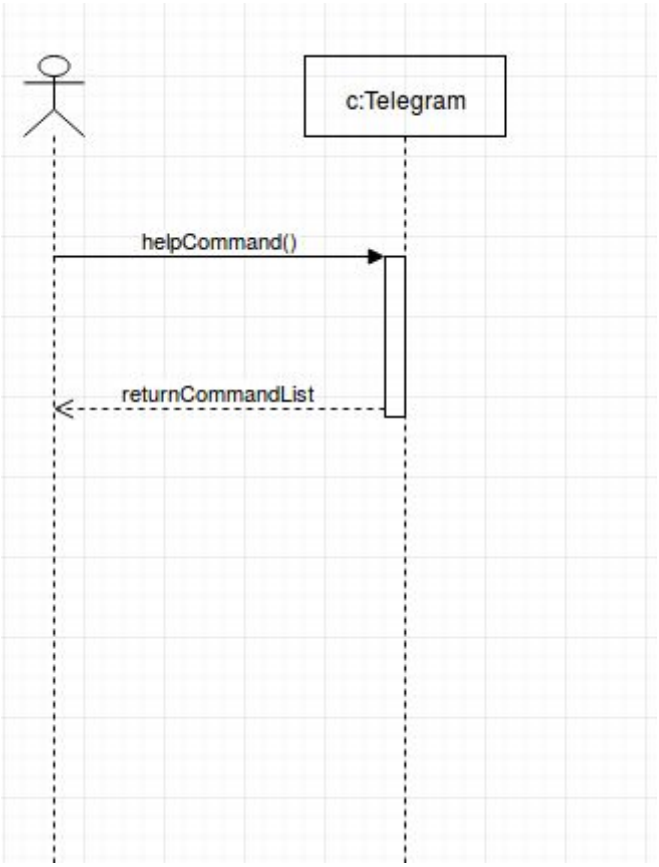
loginCommand():



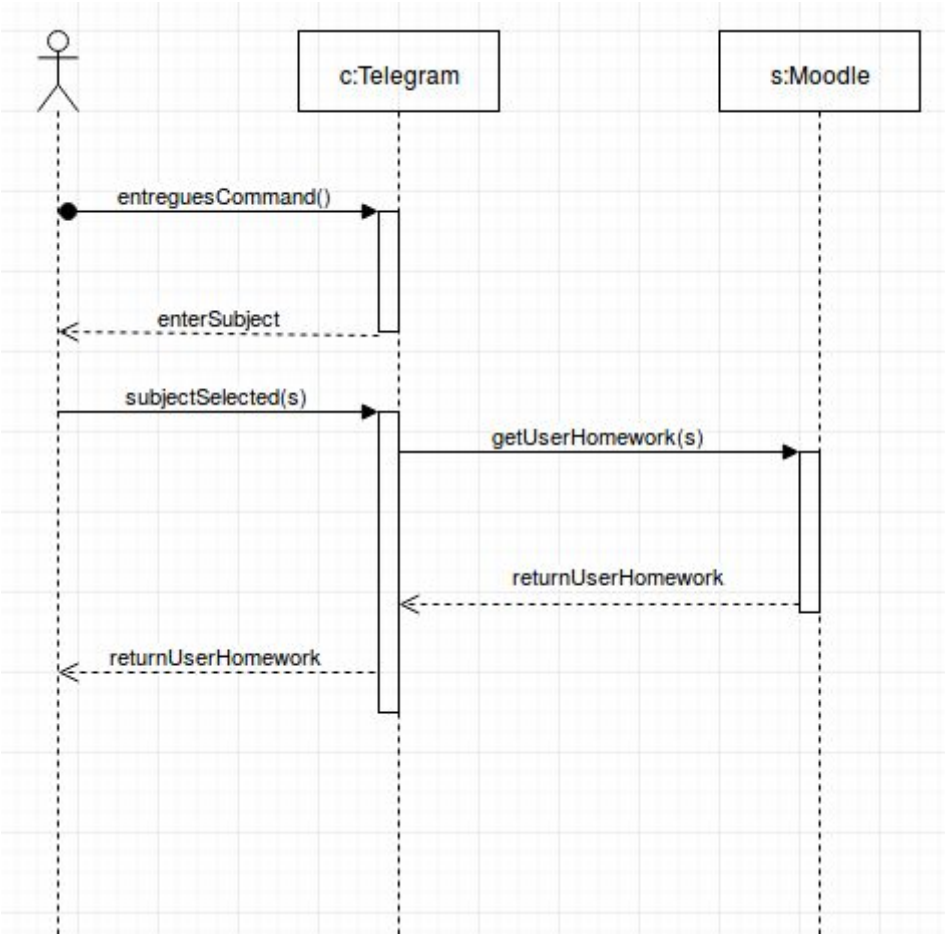
horarisCommand():



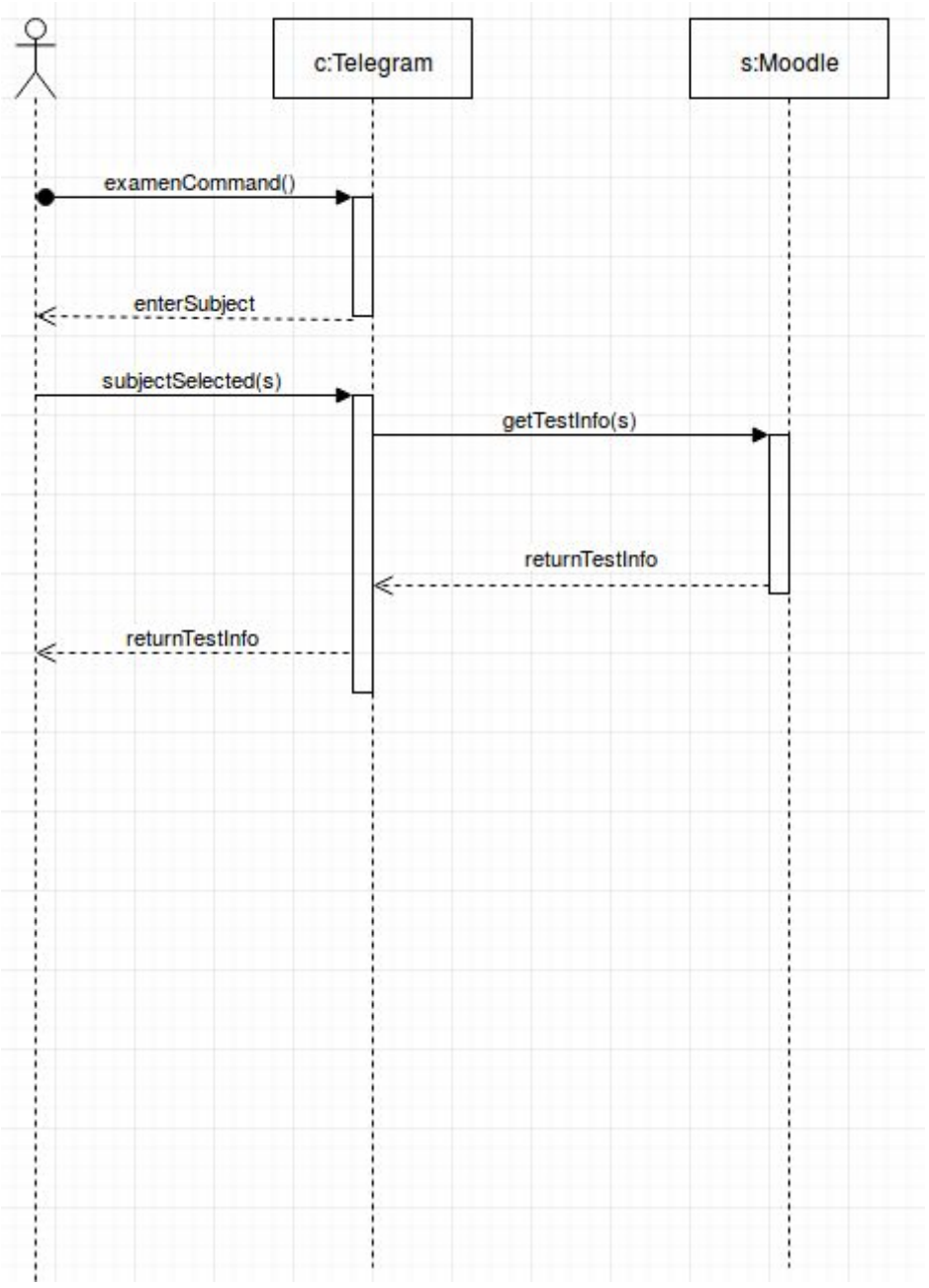
helpCommand():



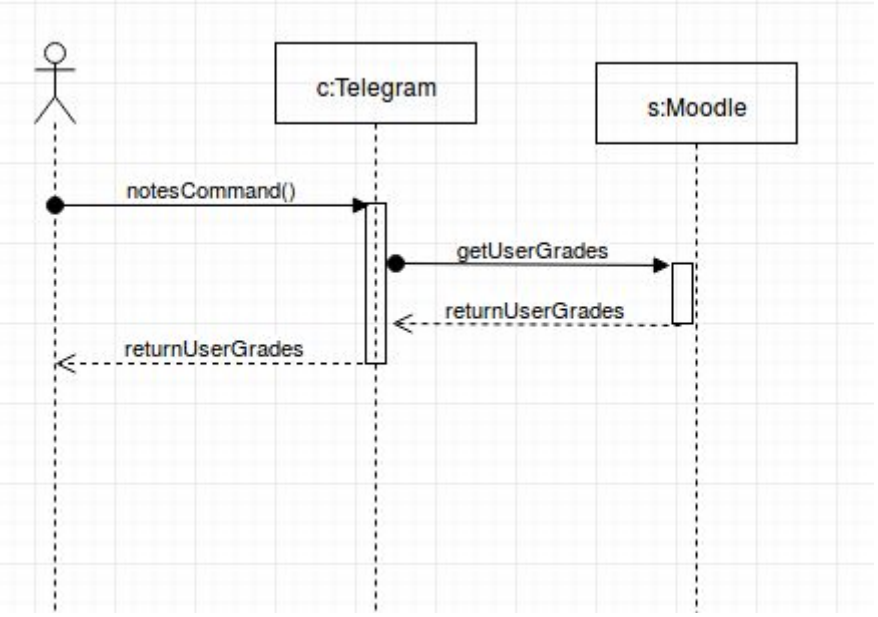
entreguesCommand():



examenCommand():

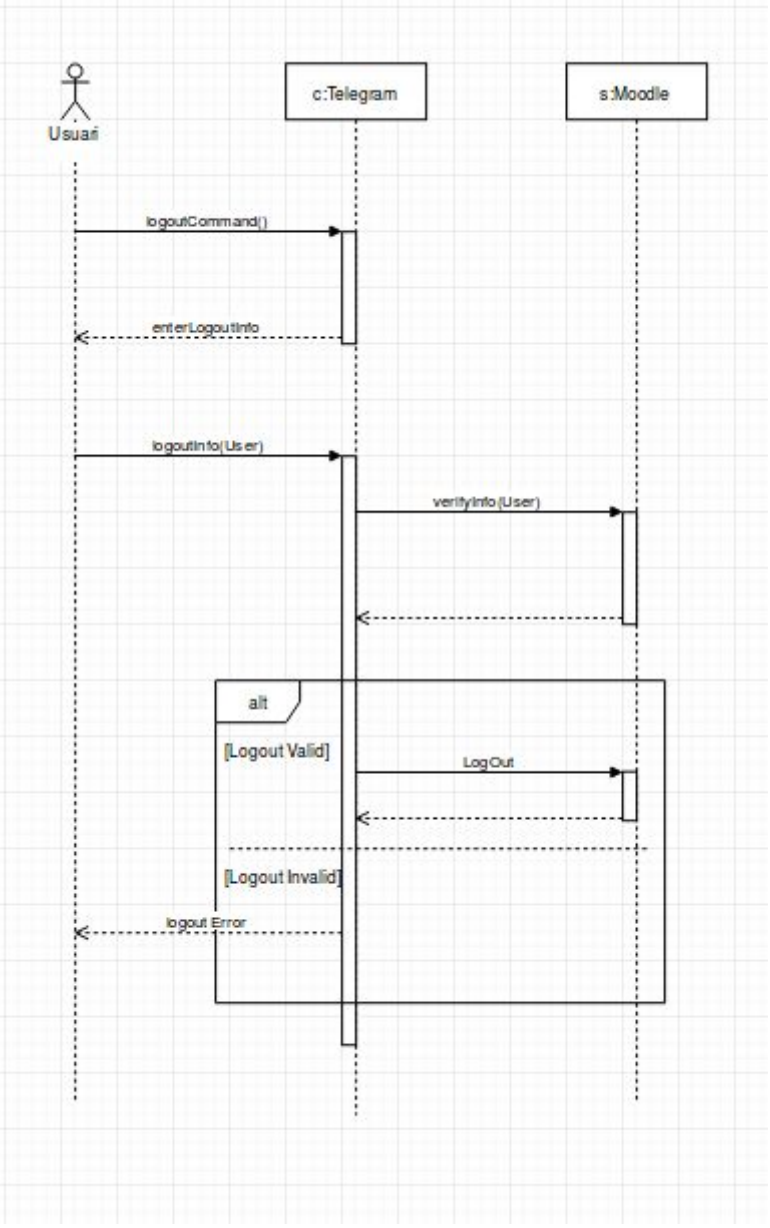


notesCommand():



idiomesCommand():

logoutCommand():



3. Instal·lació del servidor de Moodle

Per a poder instal·lar el servidor del Moodle, primer hem instal·lat al Virtual Machine una ova proporcionada per l'Institut Puig Castellar del sistema Ubuntu Server 16.04. Aquest pas l'obviarem i passarem directament a explicar les comandes que hem realitzar per al Moodle.

Pas 1: Instal·lació d'APACHE/MySQL/PHP

Obrim un terminal i introduïm el següent:

```
sudo apt-get update

sudo apt-get install apache2 mysql-client mysql-server php7.0
libapache2-mod-php7.0
```

Ens saltarà un missatge demanant un password per al root de mysql.

Pas 2: Instal·lació de software addicional

```
sudo apt-get install graphviz aspell ghostscript clamav php7.0-pspell
php7.0-curl php7.0-gd php7.0-intl php7.0-mysql php7.0-xml php7.0-xmlrpc
php7.0-ldap php7.0-zip php7.0-soap php7.0-mbstring
```

Reiniciem Apache per a que els mòduls es carreguin correctament.

```
sudo service apache2 restart
```

Instal·lem el Moodle Core Application.

```
sudo apt-get install git-core
```

Pas 3: Descarreguem el Moodle

Utilitzarem el directori /opt per a la instal·lació del Moodle 3.4. Mes endevant explicarem per què posem el Moodle Core Application en aquest directori.

```
cd /opt
sudo git clone git://git.moodle.org/moodle.git
cd moodle
```

Recuperem la llista de cada branca disponible.

```
sudo git branch -a
```

I especifiquem quina branca voldrem utilitzar.

```
sudo git branch --track MOODLE_34_STABLE origin/MOODLE_34_STABLE
```

Per finalitzar comprovem la versió de Moodle que hem especificat.

```
sudo git checkout MOODLE_34_STABLE
```

Pas 4: Copiem el respositori local a /var/www/html

```
sudo cp -R /opt/moodle /var/www/html/
sudo mkdir /var/moodledata
sudo chown -R www-data /var/moodledata
sudo chmod -R 777 /var/moodledata
sudo chmod -R 0755 /var/www/html/moodle
```

Com hem configurat un repositori local en el pas anterior, el copiarem a la nostra xarxa web després de les actualitzacions i canvis. Hem optat per deixar

el repositori local fora de la xarxa web, i ficar-lo a /opt, per a poder preparar les actualitzacions d'una manera més eficient.

Pas 5: MySQL Server

La última versió del Moodle selecciona Barracuda com a motor d'emmagatzematge predeterminat (anteriorment agafava innodb). Pero només per asegurar-nos accedirem al següent fitxer i modificarem la secció [MySQL].

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

default_storage_engine = innodb

innodb_file_per_table = 1

innodb_file_format = Barracuda
```

Actualitzem el servidor MySQL per aplicar els canvis (en cas necessari).

```
sudo service mysql restart
```

A continuació creem la base de dades del Moodle i l'usuari de Moodle MySQL.

```
mysql -u root -p


mysql>CREATE DATABASE moodle DEFAULT CHARACTER SET utf8 COLLATE
utf8_unicode_ci;

mysql>create user 'admin'@'localhost' IDENTIFIED BY 'admin';

mysql>GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,CREATE TEMPORARY
TABLES,DROP,INDEX,ALTER ON moodle.* TO admin@localhost IDENTIFIED BY
'admin';

mysql>quit;
```

Pas 6: Completem la configuració

Un cop ja tenim instal·lat el Moodle, li treiem al directori els permisos l'escriptura.

```
sudo chmod -R 0755 /var/www/html/moodle
```

A continuació obrim la següent direcció, amb la ip del nostre servidor, des d'una web:

```
http://ADREÇA.IP.DEL.SERVIDOR/moodle
```

Canviem el path per moodledata

```
/var/moodledata
```

Escollim el tipus de base de dades:

```
mysqli
```

Configuració de la base de dades:

```
Host server: localhost
Database: moodle
User: admin
Password: admin
Tables Prefix: mdl_
```

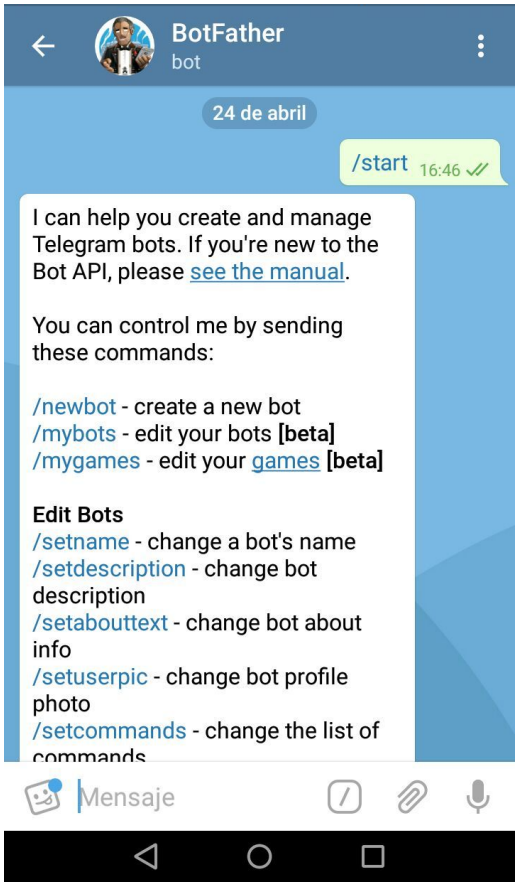
Per últim haurem de crear un administrador del Moodle que tindrà els permisos del lloc, i ja hem completat la instal·lació i configuració del nostre servidor Moodle.

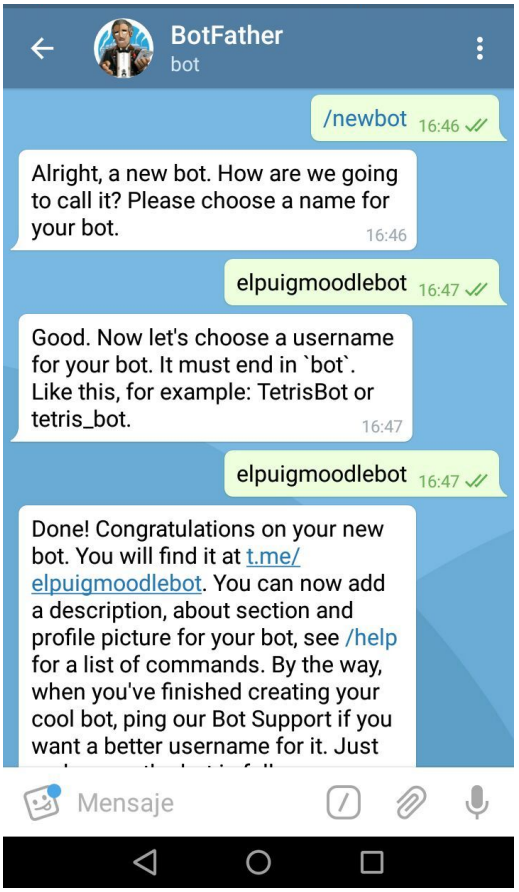
Utilitzant aquest administrador hem creat un seguit de Mòduls i alumnes, dins d'aquests mòduls hem creat varis lliurables de tipus deures i exàmens, i lis hem donat notes a cada alumne per poder, després, accedir a aquesta informació amb el Bot del Telegram.

4. La creació d'un bot a Telegram

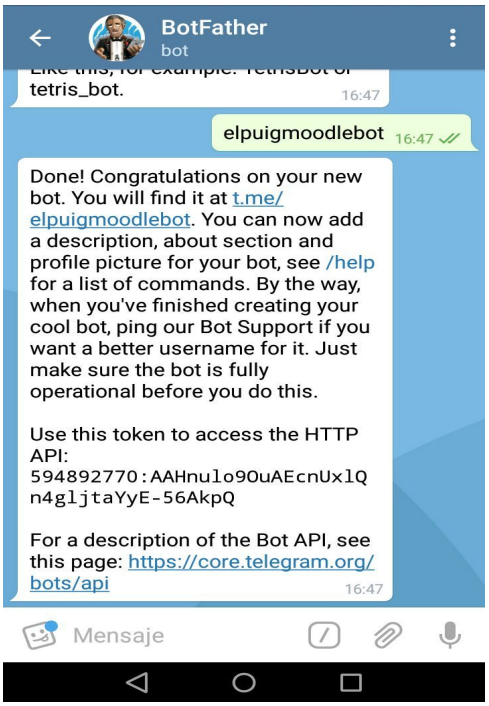
Per a la creació d'un bot de Telegram existeix una eina proporcionada per els mateixos desenvolupadors de Telegram anomenada BotFather. El BotFather és un bot que pot utilitzar l'usuari per a crear bots propis i gestionar-los. Per accedir-ne, solament s'ha de cercar al navegador de Telegram "BotFather" i iniciar una conversació.

A l'hora de crear un bot, s'ha d'indicar el nom del bot, un nom d'usuari i una breu descripció.



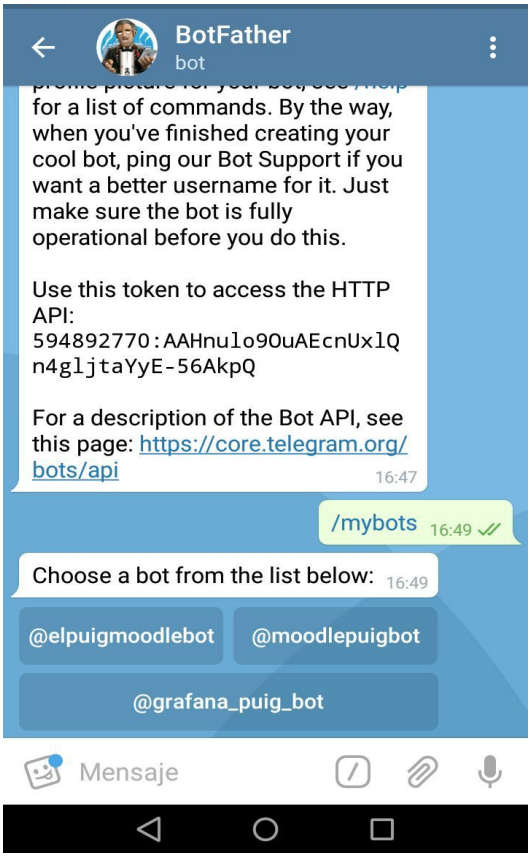


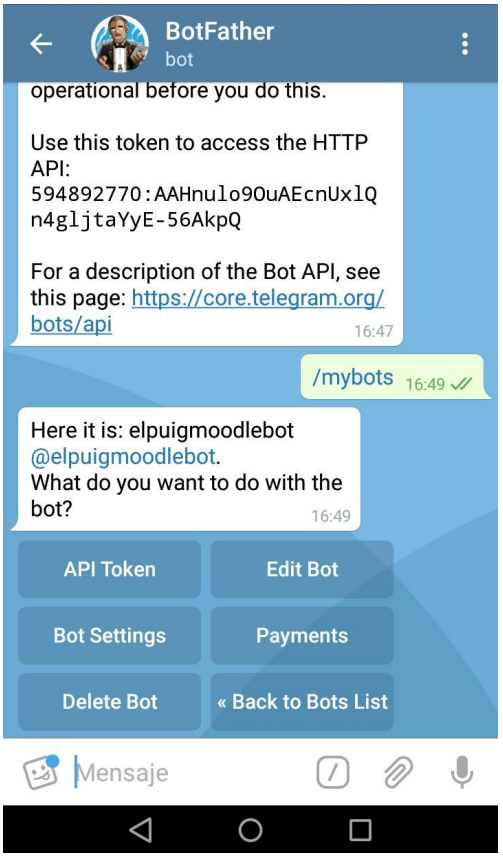
Una vegada configurades aquestes opcions BotFather ens retornarà la API del nostre bot. Aquesta API s'utiliza per vincular el codi Java amb el bot de Telegram.

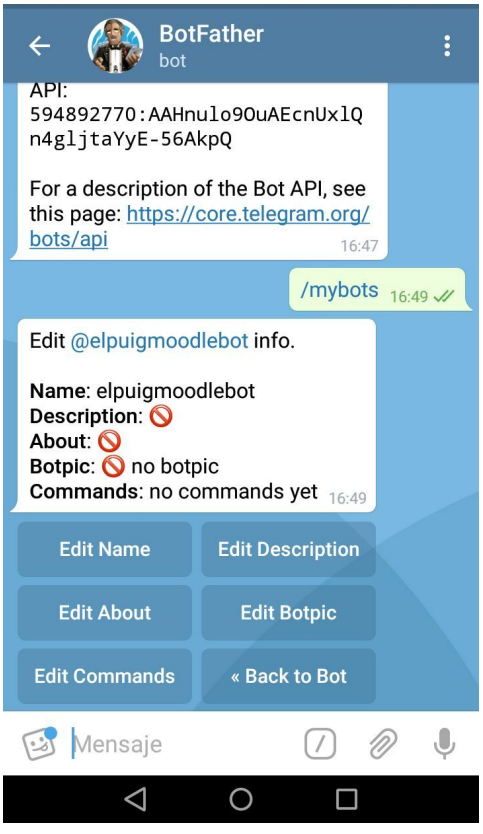


El bot està creat però òbviament, no fa res. Haurem de programar amb Java totes les seves funcionalitats. Per accedir a cada funció els bots de Telegram tenen instruccions que els usuaris utilitzaran per interactuar amb l'aplicació. Per afegir aquestes instruccions haurem d'utilitzar BotFather també.

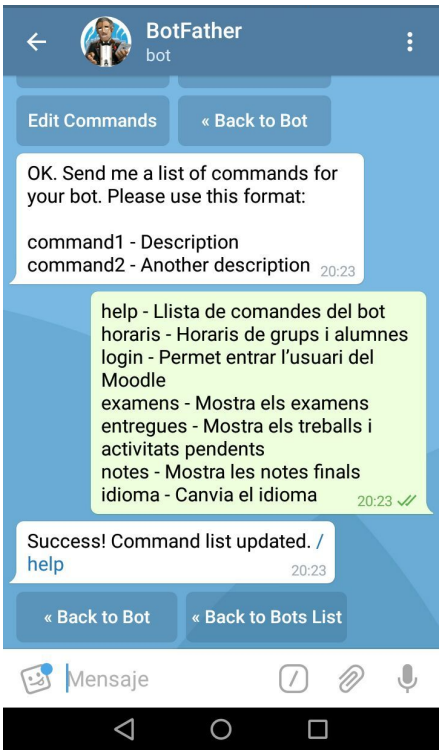
La comanda /mybots ens mostrarà una llista amb tots els bots que hem creat. Seleccionarem el del Moodle i després ens preguntarà què volem fer amb el bot. Escollirem l'opció "Edit Bot". Aquí podem modificar tota la informació del bot. Per afegir comandes és l'opció "Edit Commands".


BotFather
bot
for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
Use this token to access the HTTP API:
594892770:AAHnulo90uAEcnUxlQ n4gljtaYyE-56AkpQ
For a description of the Bot API, see this page: https://core.telegram.org/bots/api
16:47
/mybots 16:49
Choose a bot from the list below: 16:49
@elpuigmoodlebot
@moodlepuigbot
@grafana_puig_bot
Mensaje

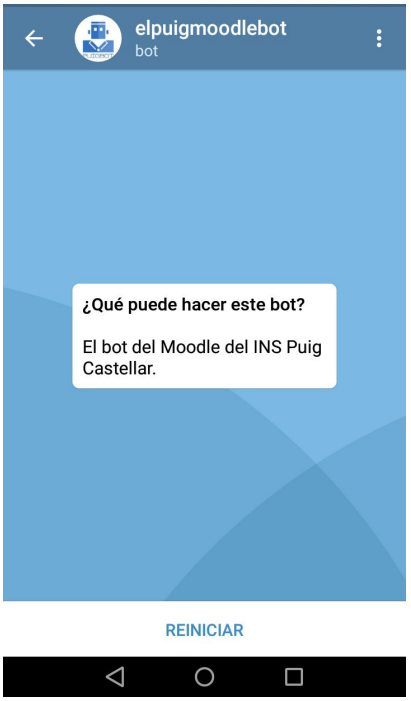

BotFather
bot
operational before you do this.
Use this token to access the HTTP API:
594892770:AAHnulo90uAEcnUxlQ n4gljtaYyE-56AkpQ
For a description of the Bot API, see this page: https://core.telegram.org/bots/api
16:47
/mybots 16:49
Here it is: elpuigmoodlebot @elpuigmoodlebot.
What do you want to do with the bot?
16:49
API Token
Edit Bot
Bot Settings
Payments
Delete Bot
« Back to Bots List
Mensaje

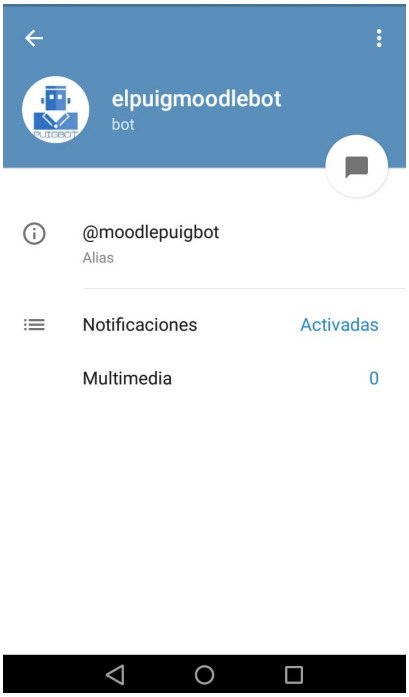

BotFather
bot
API:
594892770:AAHnulo90uAEcnUxlQ n4gljtaYyE-56AkpQ
For a description of the Bot API, see this page: https://core.telegram.org/bots/api
16:47
/mybots 16:49
Edit @elpuigmoodlebot info.
Name: elpuigmoodlebot
Description: 🚫
About: 🚫
Botpic: 🚫 no botpic
Commands: no commands yet
16:49
Edit Name
Edit Description
Edit About
Edit Botpic
Edit Commands
« Back to Bot
Mensaje

Ens demanarà una llista amb les instruccions i mostrarà el format d'entrada. Haurem d'afegir el nom i una breu descripció per cada instrucció. Les descripcions són molt importants per als usuaris ja que cada una indica que fa cada comanda.



Per últim, afegirem una foto per al bot. Això es fa amb "Edit Botpic". Tan sols hem d'enviar la imatge. Telegram s'encarregarà d'establir-la com a imatge del bot.




elpuigmoodlebot
bot
@moodlepuigbot
Alias
Notificaciones
Activadas
Multimedia
0

4.1 Codi de l'aplicació

Llibreries de Telegram

Telegram utilitza classes especials per a la programació de bots i aplicacions en Telegram. Per tant, necessitem descarregar les llibreries necessàries.

Nosaltres hem afegit les llibreries al build.gradle del nostre projecte.

build.gradle

```
dependencies {
   testCompile group: 'junit', name: 'junit', version: '4.12'
   compile 'org.telegram:telegrambots:3.6'
   compile "org.telegram:telegrambotsextensions:3.6"
   compile "org.telegram:telegrambots-abilities:3.6"
   compile "org.telegram:telegrambots-meta:3.6"

}
```

Vincular el codi Java amb el bot de Telegram

A la classe BotConfig tenim dues variables de tipus String corresponents a l'API del bot i el seu nom. Aquests valors són únics del nostre bot.

BotConfig

```
public class BotConfig {

   public static final String TOKEN_COMMAND =
"569392718:AAFSRF1Y0dgYzPpLPR7p_Foyn0z2shJ7xLk";
   public static final String USER_COMMAND = "moodlepuigbot";

}
```

Benvinguda al bot

Al accedir al bot de l'institut el primer que veiem és la descripció d'aquest. Per començar a utilitzar-lo haurem de prémer el botó "Iniciar". El bot ens donarà la benvinguda.

Tots els bots al executar-se per primera vegada utilitzen la comanda /start. Aquesta és una comanda predeterminada de Telegram que la seva única funció és iniciar el bot. En aquest cas apareixerà un missatge de benvinguda.

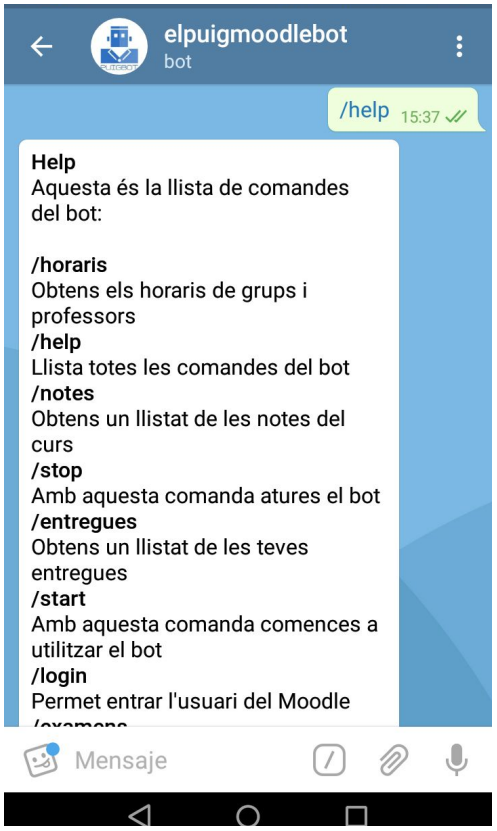
StartCommand

```
public StartCommand() {
    super("start", "Amb aquesta comanda comences a utilitzar el bot");
}

@Override
public void execute(AbsSender absSender, User user, Chat chat, String[]
arguments) {
    StringBuilder messageBuilder = new StringBuilder();

    messageBuilder.append("Benvingut "+user.getFirstName()).append("\n");
    messageBuilder.append("Aquest és el bot d'El Puig");

    SendMessage answer = new SendMessage();
    answer.setChatId(chat.getId().toString());
    answer.setText(messageBuilder.toString());

    try {
        absSender.sendMessage(answer);
    } catch (TelegramApiException e) {
        BotLogger.error(LOGTAG, e);
    }
}
```

L'ajuda del bot

L'usuari pot saber o no com funciona el bot i quines comandes té. En tot cas, cal fer una petita ajuda per a que els usuaris sàpiguen quines opcions hi ha disponibles i les instruccions necessàries per dur a terme l'acció desitjada.

La vista de l'usuari serà la següent:



La classe que gestiona la instrucció /help és HelpCommand. Aquesta comanda llistarà totes les instruccions del bot amb les seves descripcions (aquestes les agafa del llistat que s'ha establit via BotFather).

Per això és necessari que en aquesta classe hi hagi un bucle que recorri el registre de comandes i les mostri en forma de llista a l'usuari a través del MessageBuilder.

HelpCommand

```
helpMessageBuilder.append("Aquesta és la llista de comandes del bot:\n\n");

for (BotCommand botCommand : commandRegistry.getRegisteredCommands()) {
   helpMessageBuilder.append(botCommand.toString()).append("\n");
}
```

Connexió del bot amb el Moodle

L'usuari no pot consultar entregues, exàmens ni notes sense iniciar sessió. Per això cal introduir la comanda /login. La connexió amb el Moodle s'estableix quan l'usuari inicia sessió des del terminal de Telegram.

El format d'entrada per l'inici de sessió és /login seguit del usuari i contrasenya. S'hi l'usuari no introdueix ambdós arguments, el bot retornarà un missatge amb un exemple de com loggear-se. S'hi el login introduït per l'usuari és correcte se li passa a la classe MoodleAPI l'usuari de Telegram de la persona junt amb l'usuari i contrasenya del Moodle.

LoginCommand

```
if (arguments == null || arguments.length == 0) {
   messageBuilder.append(Missatges.loginHelp);

} else if (arguments.length >= 2) {

   int result = MoodleAPI.login(user.getId(), arguments[0], arguments[1]);

   if (result == 1) {
       messageBuilder.append(Missatges.loginOk);
   } else {
       messageBuilder.append(Missatges.loginError);
   }

}
```

Al mètode login de MoodleAPI establim l'accés al Moodle mitjançant la URL d'inici de sessió. Aquesta conté la IP del servidor del Moodle i els valors necessaris per loggear-se: usuari i contrasenya.

Utilitzant les variables d'usuari i contrasenya les injectarem en aquesta URL i l'enviarem al Moodle. Si les dades són correctes, tenim accés a l'usuari. Guardarem els valors token, id i e-mail de l'usuari ja que seran necessaris per realitzar consultes més tard. Totes aquestes dades es guardaran a la base de dades com a un usuari.

Si el login és correcte l'usuari iniciarà sessió sense cap problema. El bot retornarà un missatge informant a l'usuari de que el login ha sigut un èxit. En cas contrari s'enviarà un missatge d'error.

MoodleAPI

```
public static int login(int telegramId, String username, String password){
```

```
   String response = HttpUtils.get(moodleUrl + "login/token.php?username="
+ username + "&password=" + password + "&service=moodle_mobile_app");

   try {
       String token = new JSONObject(response).getString("token");

       response = HttpUtils.get(moodleUrl +
"webservice/rest/server.php?wsfunction=core_user_get_users_by_field&wstoken
=" + token + "&field=username&values[0]=" + username +
"&moodlewsrestformat=json");

       JSONObject jsonObject = new JSONArray(response).getJSONObject(0);

       String email = jsonObject.getString("email");
       String id = String.valueOf(jsonObject.getInt("id"));

       Database.get().insertUsuario(telegramId, username, token, email,
id);
   } catch (Exception e){
       return 0;
   }
   return 1;
}
```

Base de dades SQLite

És necessari emmagatzemar els usuaris que inicien sessió des del bot. D'aquesta manera no han d'estar introduint el seu usuari i contrasenya cada vegada que volen fer una acció, sinó que podem vincular el seu usuari de Telegram amb el del Moodle. També podem guardar diversa informació sobre l'usuari (el seu token, el seu id...) Per aquest motiu és necessari utilitzar una base de dades. En el nostre cas hem optat per utilitzar SQLite. Caldrà per tant instal·lar la seva llibreria, que es pot descarregar a la pàgina de SQLite Tutorial.

La classe Usuari del nostre programa és:

Usuario

```
public class Usuario {
   public int telegramId;
   public String username;
   public String token;
   public String id;
   public String email;
}
```

La classe encarregada de gestionar la base de dades s'anomena Database. Solament necessitarem una taula usuaris que contingui els usuaris de Telegram i el token corresponent a cadascun d'ells. Aquesta classe conté els

mètodes necessaris per crear la taula, inserir, seleccionar i eliminar camps i actualitzar la base de dades.

Database

```
void deleteTables(){
    try (Statement stmt = conn.createStatement()) {
        stmt.execute("DROP TABLE IF EXISTS usuarios;");
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}

void createTables(){
    try (Statement stmt = conn.createStatement()) {
        stmt.execute("CREATE TABLE IF NOT EXISTS usuarios (telegramId text,
username text, token text, email text, id text);");
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}

public void insertUsuario(int telegramId, String username, String token,
String email, String id) {
    String sql = "INSERT INTO usuarios(telegramId, username, token, email,
id) VALUES(?,?,?,?,?)";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, telegramId);
        pstmt.setString(2, username);
        pstmt.setString(3, token);
        pstmt.setString(4, email);
        pstmt.setString(5, id);
        pstmt.executeUpdate();
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}

public void deleteUsuario(String username) {
    String sql = "DELETE FROM usuarios WHERE username = ?";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, username);
        // execute the delete statement
        pstmt.executeUpdate();

    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }
}

public Usuario selectUsuarioPorTelegramId(int telegramId){
    String sql = "SELECT * FROM usuarios WHERE telegramId = ?";
```

```
    Usuario usuario = new Usuario();

    try (PreparedStatement pstmt  = conn.prepareStatement(sql)){

        pstmt.setInt(1, telegramId);
        ResultSet rs  = pstmt.executeQuery();

        while (rs.next()) {
            usuario.telegramId = telegramId;
            usuario.token = rs.getString("token");
            usuario.email = rs.getString("email");
            usuario.id = rs.getString("id");
            usuario.username = rs.getString("username");

            System.out.println("RESULTADO CONSULTA " + usuario.username);
            return usuario;
        }
    } catch (SQLException e) {
        System.out.println(e.getMessage());
    }

    return null;
}
```

Cal afegir una línia a les classes Command per a que cada vegada que executi una comanda comprovi l'usuari.

```
Usuario usuario =
Database.get().selectUsuarioPorTelegramName(user.getUserName());
```

Consultar horaris

Els horaris dels grups i professors no es troben a Moodle. Per aquest motiu la informació estarà emmagatzemada al bot. Aquest retorna una imatge de l'horari en funció del grup escollit en cas dels alumnes, o dia en cas dels professors.

Per mostrar els horaris dels diferents grups i professors la comanda és /horaris.

La vista de l'usuari és la següent:

A la classe horaris trobarem una línia que passarà a l'usuari una llista de botons, dividits en grups i professors i seguidament en els diversos grups o els dies de la setmana en cas dels professors.

```
markup.setKeyboard(Menus.MenuInlineButtonsHoraris());
```

Aquest menú es troba a la classe Menu. En aquesta classe emmagatzemarem tots els mètodes que contindran menús de tipus bot. A continuació mostrem el mètode MenuInLineButtonsHorariGrups, que mostra els diferents grups:

Menu

```
/* Crea el menu de buttons per escollir horaris dels grups */
public static List<List<InlineKeyboardButton>>
MenuInlineButtonsHorariGrups() {
   List<List<InlineKeyboardButton>> lkbGrups = new ArrayList<>();
   List<InlineKeyboardButton> row = new ArrayList<>();
   List<InlineKeyboardButton> row2 = new ArrayList<>();
   List<InlineKeyboardButton> row3 = new ArrayList<>();
   List<InlineKeyboardButton> row4 = new ArrayList<>();

   row.add(new InlineKeyboardButton()
           .setText(dataVars.GRUPS.SMX1A.toString())
           .setCallbackData(dataVars.GRUPS.SMX1A.toString()));
   row.add(new InlineKeyboardButton()
           .setText(dataVars.GRUPS.SMX1B.toString())
           .setCallbackData(dataVars.GRUPS.SMX1B.toString()));
   row.add(new InlineKeyboardButton()
           .setText(dataVars.GRUPS.SMX1C.toString())
           .setCallbackData(dataVars.GRUPS.SMX1C.toString()));
```

```
    row2.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.SMX2A.toString())
            .setCallbackData(dataVars.GRUPS.SMX1A.toString()));
    row2.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.SMX2B.toString())
            .setCallbackData(dataVars.GRUPS.SMX1B.toString()));
    row2.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.SMX2C.toString())
            .setCallbackData(dataVars.GRUPS.SMX1C.toString()));
    row3.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.GS1A.toString())
            .setCallbackData(dataVars.GRUPS.GS1A.toString()));
    row3.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.GS1B.toString())
            .setCallbackData(dataVars.GRUPS.GS1B.toString()));
    row4.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.DAM2A.toString())
            .setCallbackData(dataVars.GRUPS.DAM2A.toString()));
    row4.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.DAM2B.toString())
            .setCallbackData(dataVars.GRUPS.DAM2B.toString()));
    row4.add(new InlineKeyboardButton()
            .setText(dataVars.GRUPS.ASIX2.toString())
            .setCallbackData(dataVars.GRUPS.ASIX2.toString()));
    row3.add(new InlineKeyboardButton()
            .setText("<---")
            .setCallbackData("<---"));
    lkbGrups.add(row);
    lkbGrups.add(row2);
    lkbGrups.add(row4);
    lkbGrups.add(row3);

    return lkbGrups;

}
```

Al fer clic en el grup o dia (en cas dels professors), el bot retornarà una imatge amb l'horari. El codi que realitza aquesta operació es troba a la classe ElPuigMoodleBot.

ElPuigMoodleBot

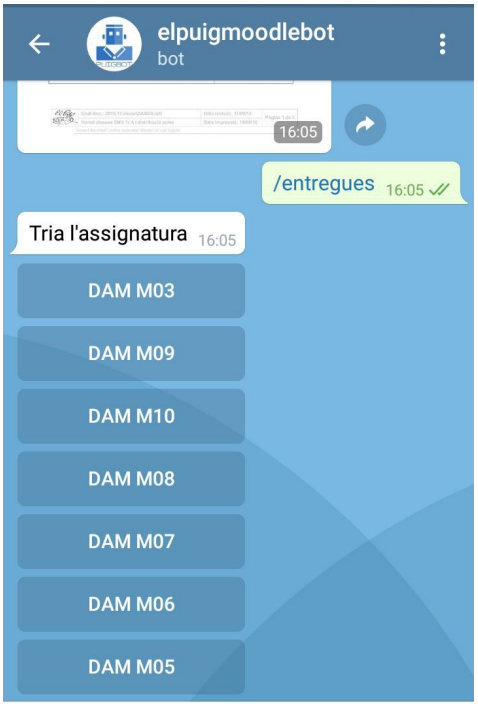
```
switch (tria) {
    case "Professors":
        markup.setKeyboard(Menus.MenuInlineButtonsHorarisProfes());
        enviarResposta(callbackQuery, markup, "Tria els horaris de:");
        break;
    case "Grups":
        markup.setKeyboard(Menus.MenuInlineButtonsHorariGrups());
        enviarResposta(callbackQuery, markup, "Tria el grup:");
        break;
    case "<---":
        markup.setKeyboard(Menus.MenuInlineButtonsHoraris());
        enviarResposta(callbackQuery, markup, "Tria de qui vols veure
```

```
horaris:");
        break;
    case "Dilluns": enviarResposta(answerPhoto, dataVars.HPDilluns); break;
    case "Dimarts": enviarResposta(answerPhoto, dataVars.HPDimarts); break;
    case "Dimecres":    enviarResposta(answerPhoto, dataVars.HPDimecres);
break;
    case "Dijous":  enviarResposta(answerPhoto, dataVars.HPDijous); break;
    case "Divendres":   enviarResposta(answerPhoto, dataVars.HPDivendres);
break;
    case "SMX1A" :  enviarResposta(answerPhoto,dataVars.HSMX1A); break;
    case "SMX1B":   enviarResposta(answerPhoto,dataVars.HSMX1B); break;
    case "SMC1C":   enviarResposta(answerPhoto,dataVars.HSMX1C); break;
    case "SMX2A":   enviarResposta(answerPhoto,dataVars.HSMX2A); break;
    case "SMX2B":   enviarResposta(answerPhoto,dataVars.HSMX2B); break;
    case "SMX2C":   enviarResposta(answerPhoto,dataVars.HSMX2C); break;
    case "GS1B":    enviarResposta(answerPhoto,dataVars.HGS1B); break;
    case "GS1A":    enviarResposta(answerPhoto,dataVars.HGS1A); break;
    case "ASIX2":   enviarResposta(answerPhoto,dataVars.HASIX2A); break;
    case "DAM2A":   enviarResposta(answerPhoto,dataVars.HDAM2A); break;
    case "DAM2B":   enviarResposta(answerPhoto,dataVars.HDAM2B); break;
    ...
    ...
    ...
}
```

Consultar entregues i exàmens

Un dels objectius principals del projecte es que l'usuari pugui accedir als seus treballs i entregues de Moodle. La comanda que realitza aquesta acció és /entregues.

La vista de l'usuari serà la següent:



De la mateixa manera, l'usuari serà capaç de consultar els seus exàmens amb la comanda /examens. La vista serà la següent:

Encara que totes dues comandes mostren diferent informació, ambdues tenen similituds, ja que els examen són si més no entregues també. Aquesta similitud s'aplica també al codi. A continuació explicarem com realitzar consultes sobre les entregues o *assigments* de Moodle.

Extreure informació

Per fer aquesta consulta haurem d'accedir primerament als cursos i després a les entregues de cadascun d'ells.

Serà necessari per tant crear classes per a cursos i entregues. En aquestes haurem d'afegir tots els atributs que volem guardar d'aquests dos objectes.

Course

```
public class Course {
   public String id;
   public String shortname;
   public String fullname;
}
```

Entrega

```
public class Entrega {
   public int id;
   public String nom;
   public boolean entregada;
   public int duedate;
   public String grade;
   public String linkNotes;
}
```

A MoodleAPI trobarem el mètode getCourses que és el que cercarà tots els cursos de l'usuari. A través de l'idenficador de l'usuari (el id de Telegram en aquest cas) es comprova si l'usuari existeix a la base de dades. Seguidament, és realitza la consulta sobre el Moodle per poder obtenir tots els cursos.

La sortida de les dades és realitza en format JSON. Per podem emagatzemar les dades s'ha de fer un *parsing* del document. Guardarem tots els cursos en un ArrayList de cursos. De cada curs guardarem el seu id, el nom curt i el nom llarg. Aquest atributs són necessaris per poder mostrar a l'usuari informació sobre el curs.

MoodleAPI

```
public static List<Course> getCourses(int telegramId){

   Usuario usuario = Database.get().selectUsuarioPorTelegramId(telegramId);

   String response = HttpUtils.get(moodleUrl +
"webservice/rest/server.php?wsfunction=core_enrol_get_users_courses&wstoken
="+usuario.token+"&userid="+usuario.id+"&moodlewsrestformat=json");


   List<Course> courses = new ArrayList<>();

   new JSONArray(response).forEach(item -> {
       JSONObject courseJSON = (JSONObject) item;

       Course course = new Course();
       course.fullname = courseJSON.getString("fullname");
       course.shortname = courseJSON.getString("shortname");
       course.id = String.valueOf(courseJSON.getInt("id"));
```

```
        courses.add(course);

    });

    return courses;

}
```

Per aconseguir el llistat d'entregues utilitzarem el mètode getEntregues. És necessari passar-li al mètode l'identificador de l'usuari i a més a més l'identificador del curs, ja que no podrem accedir a les entregues d'un curs sense el seu id.

El procediment per treure les dades de les entregues és molt semblant al que hem vist anteriorment. Realitzarem la consulta que treu informació sobre les entregues, afegint el id del curs. Haurem de recorrer el document JSON fins arribar a l'apartat on apareixen les entregues. Aquest s'anomena assigments. Una vegada aquí, farem un *parsing* guardant l'identificador, el nom i la data d'entrega.

MoodleAPI

```
public static List<Entrega> getEntregues(int telegramId, String courseId){

    Usuario usuario = Database.get().selectUsuarioPorTelegramId(telegramId);

    String response = HttpUtils.get(moodleUrl +
"webservice/rest/server.php?wsfunction=mod_assign_get_assignments&wstoken="
+usuario.token+"&courseids[0]="+courseId+"&moodlewsrestformat=json");

    List<Entrega> entregues = new ArrayList<>();

    JSONArray entreguesJSON = new
JSONObject(response).getJSONArray("courses").getJSONObject(0).getJSONArray(
"assignments");

    entreguesJSON.forEach(item -> {
        JSONObject entregaJSON = (JSONObject) item;

        Entrega entrega = new Entrega();
        entrega.id = entregaJSON.getInt("id");
        entrega.nom = entregaJSON.getString("name");
        entrega.duedate = entregaJSON.getInt("duedate");
        entregues.add(entrega);

    });
    ...
    ...
    ...


    }
```

També volem mostrar a l'usuari la nota de cada entrega i s'hi ha estat entregada o no, però aquestes dades no apareixen a la consulta realitzada anteriorment. Realitzarem una segona consulta sobre el document que mostra els detalls d'una entrega. Per això és necessari el token de l'usuari i l'id de la entrega en concret.

Consultarem els camp submission del document JSON per determinar si l'actividad ha estat entregada o no. Aquest camp serà boleà.

De la mateixa manera comprovarem el camp *grade* que és el que determina si una activitat ha estat calificada o no. Si aquest camp no existeix vol dir que no hi ha valoració.

MoodleAPI

```
public static List<Entrega> getEntregues(int telegramId, String courseId){
    ...
    ...
    ...
//Per cada entrega agafem l'id de la entrega i amb l'id busquem el grade
    for (Entrega entrega : entregues) {
        String response2 = HttpUtils.get(moodleUrl +
"webservice/rest/server.php?wsfunction=mod_assign_get_submission_status&wst
oken="+usuario.token+"&assignid="+entrega.id +"&moodlewsrestformat=json");

        try {
            new
JSONObject(response2).getJSONObject("lastattempt").getJSONObject("submissio
n");

            entrega.entregada = true;
        } catch (Exception e){
            entrega.entregada = false;
        }

        try {
            //Si no hi ha feedback, es que no està calificada
            entrega.grade = new
JSONObject(response2).getJSONObject("feedback").getJSONObject("grade").getS
tring("grade");
        } catch (Exception e){
            entrega.grade = "Sense calificar";
        }

    }

    return entregues;
}
```

Mostrar assignatures en forma de menú

Ja tenim dos mètodes que extreuen tota la informació necessària del Moodle. A continuació haurem de mostrar aquesta informació a l'usuari.

Primerament mostrar un menú de botons amb totes les assignatures. L'usuari en prémer en aquests botons tindrà accés al llistat d'entregues o exàmens. Aquest menú el crearem a la classe Menus.

Crearem una llista de botons, i cridarem al mètode getCoruses de MoodleAPI. Per cada curs afegirem un botó amb el seu shortname.

Farem aquesta acció per a tots els cursos excepte el de tutoria, ja que aquest és un curs especial on solament es troben les notes finals i no volem mostrar-lo al llistat.

Menus

```
/* Crea el menu de buttons per escollir asignatures */
public static List<List<InlineKeyboardButton>>
MenuInlineButtonsAssignatures(String tipo, User user) {
   List<List<InlineKeyboardButton>> lkb = new ArrayList<>();

   List<Course> cursos = MoodleAPI.getCourses(user.getId());

   for (Course course : cursos) {
       if(course.fullname.toLowerCase().contains("tutoria")){

       }
       else {
           List<InlineKeyboardButton> row = new ArrayList<>();
           row.add(new InlineKeyboardButton()
                   .setText(course.shortname)
                   .setCallbackData(tipo + ":" + course.id + ":" +
user.getId()));
           lkb.add(row);
       }
   }

   return lkb;

}
```

Per cridar a aquest menú ho haurem de fer a les classes EntreguesCommand i ExamenCommand. El primer valor és el tipus (entregues o exàmens) i el segon valor és l'usuari.

EntreguesCommand

```
InlineKeyboardMarkup markup = new InlineKeyboardMarkup();
markup.setKeyboard(Menus.MenuInlineButtonsAssignatures("entregues", user));
```

ExamenCommand

```
InlineKeyboardMarkup markup = new InlineKeyboardMarkup();
markup.setKeyboard(Menus.MenuInlineButtonsAssignatures("examen", user));
```

Mostrar informació a l'usuari

Ja tenim el menú, però hem de mostrar els llistats d'entregues i exàmens. Això és fa a la classe ElPuigMoodleBot. Com hem vist a l'apartat d'horaris hi ha un *switch* que mostra tota la informació relativa als horaris. Crearem un *default* en aquest *switch* on mostrarem la informació de les entregues.

Utilitzant *.split* determinarem el tipus de data que hem de mostrar, obtindrem el id del curs i l'usuari. Amb aquesta informació podem determinar quines dades mostrar a través de sentències condicionals.

ElPuigMoodleBot

```
switch (tria) {
   ...
   ...
   ...
   default:
       String[] partsTria = tria.split(":");
       String tipus = partsTria[0]; // tipus: entregues, examen, notes
       String courseId = partsTria[1];
       int uid = Integer.parseInt(partsTria[2]);

       if (tipus.equals("entregues")){
           System.out.println("Mostrant entregues...");
           respondreText(answer, "Aquestes són les entregues: \n" +
buildStringEntregues(uid, courseId));
       }

       if (tipus.equals("examen")){
           System.out.println("Mostrant examens...");
           respondreText(answer, "Aquests són els examens: \n" +
buildStringExamens(uid, courseId));
       }
}
```

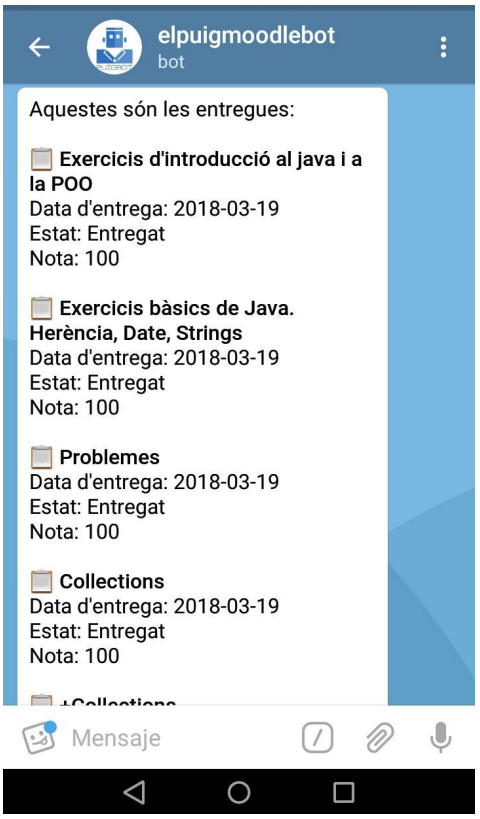
Per mostrar les entregues cridarem al mètode buildStringEntregues. Aquest mètode crida a getEntregues de MoodleAPI per obtenir totes les entregues. Per cada entrega comprova que el nom no conté “examen” o “prova”. En aquest cas no farà res. Per la resta d’entregues farà les següents accions:

- Formatarà la data de l’entrega de manera que apareixerà amb la sortida: dd/mm/aa.
- Formatarà la nota de manera que apareixeran números sencers.
- Comprovarà l’estat de l’entrega: entregat o no entregat.
- Afegira al string el nom de l’entrega, la data formatada, l’estat i la nota formatada.

ElPuigMoodleBot

```
/* Genera una cadena de text amb totes les entregues */
String buildStringEntregues(int telegramId, String courseId){

   List<Entrega> entregues = MoodleAPI.getEntregues(telegramId, courseId);
   StringBuilder sb = new StringBuilder();

   for(Entrega entrega : entregues){
       System.out.println(entrega.nom + entrega.id);
       if (entrega.nom.toLowerCase().contains("examen") ||
entrega.nom.toLowerCase().contains("prova"))  {

       }
       else {
           Instant data = Instant.ofEpochSecond(entrega.duedate);
           String dataString = data.toString();
           String[] dataFormat = dataString.split("T");
           String[] gradeFormat = entrega.grade.split(Pattern.quote("."));

           String estat;
           if (entrega.entregada == true) {
               estat = "Entregat";
           } else {
               estat = "No entregat";
           }

           sb.append("\n<b>" + Emoji.CLIPBOARD + " " + entrega.nom + "</b>
\n");
           sb.append("Data d'entrega: " + dataFormat[0] + " \n");
           sb.append("Estat: " + estat + " \n");
           sb.append("Nota: " + gradeFormat[0] + " \n");
       }
   }
   return sb.toString();
}
```

Al executar aquest mètode l'usuari veurà en pantalla el llistat de les entregues amb tota la informació adient.



Per mostrar els exàmens cridarem al mètode buildStringExamens. És molt semblant al mètode anterior. Per cada entrega formatarà la data i la nota, comprovarà l'estat per veure si l'examen ha estat realitzat i solament mostrarà les entregues que contenen "examen" o "prova" al seu nom.

ElPuigMoodleBot

```
/* Genera una cadena de text amb tots els examens */
String buildStringExamens(int telegramId, String courseId){

   List<Entrega> entregues = MoodleAPI.getEntregues(telegramId, courseId);
   StringBuilder sb = new StringBuilder();

   for(Entrega entrega : entregues) {
       Instant data = Instant.ofEpochSecond(entrega.duedate);
       String dataString = data.toString();
       String[] dataFormat = dataString.split("T");
       String[] gradeFormat = entrega.grade.split(Pattern.quote("."));

       String estat;
       if (entrega.entregada == true) {
           estat = "Realitzat";
       } else {
```

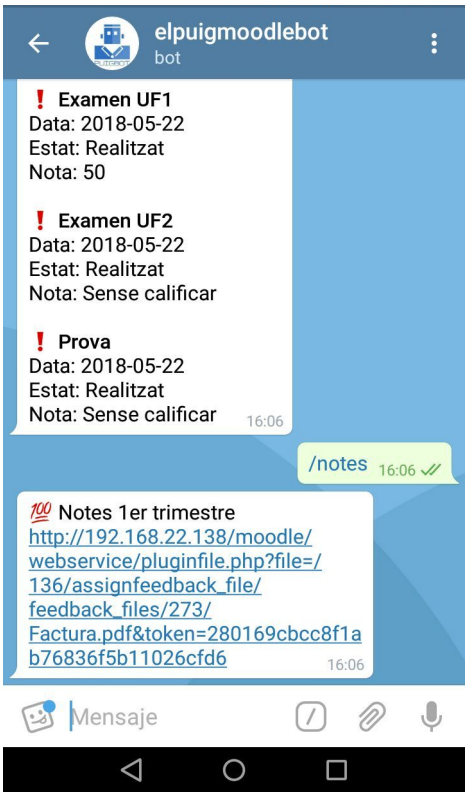
```
            estat = "No realitzat";
        }

        if (entrega.nom.toLowerCase().contains("examen") ||
entrega.nom.toLowerCase().contains("prova")) {
            sb.append("\n<b>" + Emoji.HEAVY_EXCLAMATION_MARK_SYMBOL + " " +
entrega.nom + "</b> \n");
            sb.append("Data: " + dataFormat[0] + " \n");
            sb.append("Estat: " + estat + " \n");
            sb.append("Nota: " + gradeFormat[0] + " \n");
        }
    }
    return sb.toString();
}
```

Al executar aquest mètode la vista de l'usuari serà la següent:

Consultar notes

El bot del Puig també pot mostrar als usuaris les seves notes finals. El que fa és retornar a l'usuari enllaços a un arxiu de tipus *.pdf* amb els butlletins de notes dels diferents trimestres.

La comanda que s'encarrega de mostrar les notes finals és /notes. La vista de l'usuari serà la següent:



Per gestionar aquestes accions al codi font trobem la classe NotesCommand. Aquesta classe consulta els cursos amb el mètode getCourses de MoodleAPI i emmagatzema només l'id del curs de tutoria. Això ho fa perquè el curs tutoria és un curs especial que conté les notes finals.

Una vegada té l'identificador del curs de tutoria crida al mètode getNotes. Aquest retornarà un nom i un string que mostrarem a l'usuari.

NotesCommand

```
...
...
    String tutoriaId = "";
    for(Course course: MoodleAPI.getCourses(user.getId())){
        if(course.fullname.toLowerCase().contains("tutoria")){
            tutoriaId = course.id;
            break;
        }
    }

    for(Entrega entrega : MoodleAPI.getNotes(user.getId(), tutoriaId)){
        if(entrega.nom.contains("Notes") ) {
            messageBuilder.append("\n" + Emoji.HUNDRED_POINTS_SYMBOL + "
```

```
" + entrega.nom + " \n");
                messageBuilder.append(entrega.linkNotes + " \n\n");
            }
        }
    ...
    ...
}
```

El mètode getNotes és gairebé igual a getEntregues. Realitzem la consulta sobre el curs de tutoria, emmagatzemant la informació necessària de cada entrega (id, nom i data).

Una vegada fet això, per cada entrega hem de consultar el camp fileurl que és el que conté l'enllaç de l'arxiu *.pdf*. Una vegada obtingut el camp haurem d'inserir el token de l'usuari al enllaç, sinó el link no funcionarà i l'usuari no es podrà descarregar l'arxiu *.pdf*.

MoodleAPI

```
public static List<Entrega> getNotes(int telegramId, String courseId){

   Usuario usuario = Database.get().selectUsuarioPorTelegramId(telegramId);

   String response = HttpUtils.get(moodleUrl +
"webservice/rest/server.php?wsfunction=mod_assign_get_assignments&wstoken="
+usuario.token+"&courseids[0]="+courseId+"&moodlewsrestformat=json");

   List<Entrega> entregues = new ArrayList<>();

   JSONArray entreguesJSON = new
JSONObject(response).getJSONArray("courses").getJSONObject(0).getJSONArray(
"assignments");

   entreguesJSON.forEach(item -> {
       JSONObject entregaJSON = (JSONObject) item;

       Entrega entrega = new Entrega();
       entrega.id = entregaJSON.getInt("id");
       entrega.nom = entregaJSON.getString("name");
       entrega.duedate = entregaJSON.getInt("duedate");
       entregues.add(entrega);

   });

   for (Entrega entrega : entregues) {
       String response2 = HttpUtils.get(moodleUrl +
"webservice/rest/server.php?wsfunction=mod_assign_get_submission_status&wst
oken="+usuario.token+"&assignid="+entrega.id +"&moodlewsrestformat=json");

       try {
           entrega.linkNotes = new
JSONObject(response2).getJSONObject("feedback").getJSONArray("plugins").get
```

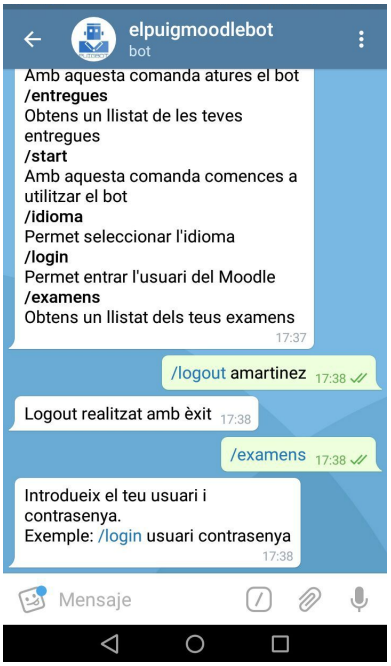
```
JSONObject(2).getJSONArray("fileareas").getJSONObject(0).getJSONArray("file
s").getJSONObject(0).getString("fileurl");
            entrega.linkNotes = entrega.linkNotes.replace("pluginfile.php",
"pluginfile.php?file=");
            entrega.linkNotes += "&token=" + usuario.token;
            System.out.println(entrega.linkNotes);
        } catch (Exception e){
            entrega.linkNotes = "No disponibles";
        }
    }

    return entregues;
}
```

Tancar sessió

També hem d'implementar la opció de tancar sessió per si l'usuari desitja loggear-se amb un altre usuari. La comanda que fa aquesta acció és /logout seguida del nom d'usuari de Moodle. Si el format d'entrada no és correcte el bot retornarà un missatge d'ajuda per a l'usuari.

Si l'usuari intenta accedir a les entregues, exàmens o notes sense haver iniciat sessió sortirà un missatge d'ajuda.

La vista de l'usuari serà la següent:



La classe encarrega de comprovar el tancament de sessió es LogOutCommand. Aquesta comprova que els paràmetres introduits per l'usuari

són els correctes. Després cridarà al mètode logout de MoodleAPI. Segons el *return* el tancament de sessió serà correcte o no.

LogOutCommand

```
 ...
   ...
if (arguments == null || arguments.length == 0) {
   messageBuilder.append(Missatges.Idioma.logoutHelp);

} else if (arguments.length >= 1) {

   int result = MoodleAPI.logout(user.getId(), arguments[0]);

   if (result == 1) {
       messageBuilder.append(Missatges.Idioma.logoutOk);

   } else {
       messageBuilder.append(Missatges.Idioma.logoutError);
   }
}
 ...
   ...
```

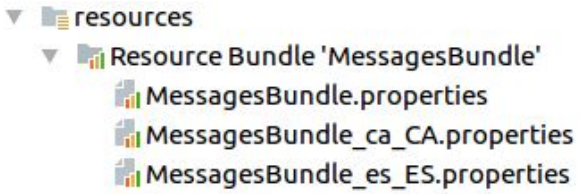
El mètode logout agafa el telegram id i l'username de l'usuari. Aquests serveixen per identificar l'usuari que tancarà sessió. Després cridarà al mètode deleteUsuari de la base de dades. D'aquesta manera l'usuari queda esborrat de la BD del programa, tancant la seva sessió.

MoodleAPI

```
public static int logout(int telegramId, String username){
   try {
       Database.get().deleteUsuario(username);
   } catch (Exception e){
       return 0;
   }
   return 1;
}
```

Idiomes

És possible afegir diferents idiomes a la aplicació. Per això s'hauran de traduir tots els missatges que es mostren a l'usuari en fitxers de recursos. Es crearà un fitxer per cada idioma a traduir. La nostra aplicació té dos idiomes disponibles: català i castellà.

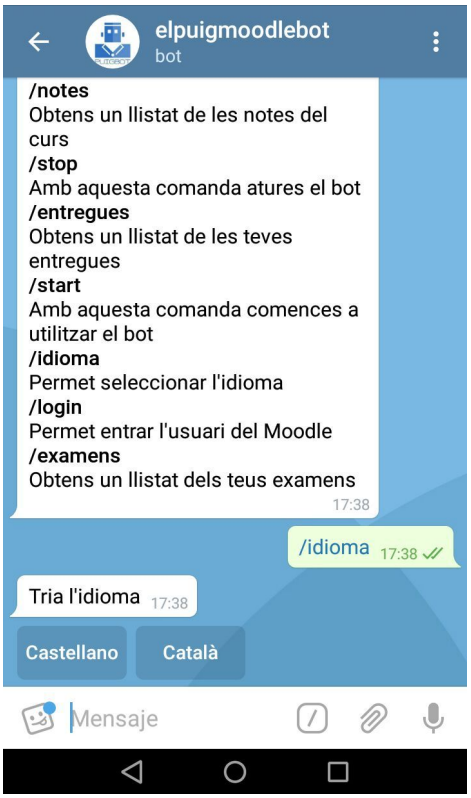


A continuació es mostra un exemple del fitxer de català.

MessagesBundle_ca_CA.properties

```
loginActive = Ja estàs loggeat
loginHelp = Introdueix el teu usuari i contrasenya. \nExemple: /login
usuari contrasenya
loginOk = Login realitzat amb èxit
loginError = Login incorrecte
triaAsignatura = Tria de quina assignatura vols veure l'entrega
LlistaComandes = Llista totes les comandes del bot
LlistaComandes2 = Aquesta és la llista de comandes del bot
LlistaEntregas = Obtens un llistat de les teves entregues
ComandesBot = Llista totes les comandes del bot
Veurehoraris = Tria de qui vols veure horaris
selectIdioma  Permet entrar l'usuari del Moodle
TriaIdioma = Tria l'idioma
LlistatNota = Llistat de notes
BotDeProva = Aquest és el bot de proves de El Puig
ComençaUtilBot= Amb aquesta comanda comences a utilitzar el bot
AturaBot = Amb aquesta comanda atures el bot
ComandaNoImplementada = La comanda '%s' encara no està implementada o no és
d'aquest bot. Aquí tens una ajuda %s
ElteuMissatge = El teu missatge
horaris = Tria els horaris de
TriaGrup = Tria el grup
VeureHoraris = Tria de qui vols veure horaris
Benvingut = Benvingut
Adeu = Adeu
```

La comanda encarregada de canviar l'idioma del bot és /idiomes. L'usuari veurà un menú amb els idiomes disponibles i haurà de seleccionar un d'ells.



La classe IdiomaCommand cridarà al menu amb la llista d'idiomes.

IdiomaCommand

```
messageBuilder.append("Tria l'idioma").append("\n");

InlineKeyboardMarkup markup = new InlineKeyboardMarkup();
markup.setKeyboard(Menus.MenuInlineButtonsIdioma());
```

En la classe Menu trovem el mètode MenuInlineButtonsIdioma que crea el menú d'idiomes on estan establits tots els llenguatges disponibles.

Menus

```
public static List<List<InlineKeyboardButton>> MenuInlineButtonsIdioma() {
   List<List<InlineKeyboardButton>> lkbIdioma2 = new ArrayList<>();
   List<InlineKeyboardButton> row = new ArrayList<>();

   row.add(new InlineKeyboardButton()
           .setText("Castellano")
           .setCallbackData("Castellano"));
   row.add(new InlineKeyboardButton()
```

```
            .setText("Català")
            .setCallbackData("Català"));
    lkbIdioma2.add(row);

    return lkbIdioma2;
}
```

Per últim, a la classe ElPuigMoodleBot afegirem les opcions dels idiomes al *switch*. D'aquesta manera quan l'usuari premi a un botó, es cridarà al mètode establirIdioma que s'encarrega de establit el llenguatge preferit per l'usuari.

ElPuigMoodleBot

```
  ...
    ...
switch (tria) {
    case "Català":
        markup.setKeyboard(Menus.MenuInlineButtonsIdioma());
        establirIdioma("catala");
        break;
    case "Castellano":
        markup.setKeyboard(Menus.MenuInlineButtonsIdioma());
        establirIdioma("castellano");
        break;
  ...
    ...

void establirIdioma(String idioma){
    if(idioma.equals("catala")){
        Missatges.seleccionarIdioma("ca", "CA");
    } else {
        Missatges.seleccionarIdioma("es", "ES");
    }
}
```

5. Codi API Moodle

Per poder accedir a la informació dels alumnes que tenim al Moodle, podem fer-ho de dues maneres, a través del webservice que ens proporciona el Moodle o a través de comandes des d'el terminal. A continuació introduïm varies captures i explicacions de com les hem utilitzat, i com hem pogut accedir finalment a la informació dels alumnes del Moodle.

El resultat d'aquestes comandes (tant del webservice com per terminal) és la informació que nosaltres necessitarem per enviar-li al Bot de Telegram quan ens la demani.

¿Qué és el Webservice?

Moodle ens ofereix una serie de funcionalitats (gestió d'alumnes i cursos, configuració de rols, pujada i baixada d'arxius, fotos, etc.) que son interessants poder utilitzar-les amb altres entorns, en el nostre cas, amb una aplicació móvil. Aquesta es la principal funcionalitat dels Webservice que Moodle proporciona, així com la integració d'aplicacions externes dintre del Moodle a través de IMS LTI, suposant que en un futur es podrien afegir noves funcionalitats per a aquest servei.

Activem el WebService.

El primer que hem de fer es activar els webservice. Per fer això haurem d'anar al nostre Moodle i seguir els següents pasos:
Home-> Site administration-> Advanced features
Marquem la opció **Enable web services** i guardem.
Ara anem a :
Home-> Site administration-> Plugins->Webs ervices->Manage protocols.
Aqui activem la opcio **XML-RPC** i guardem.

Creem un WebService.

Anem a:

Home->Site administration->Plugins->Web Services->External services
cliquem a **Add** i especifiquem un nom (per exemple, admin_test) i marquem la opció **Enabled**.

Ara tenim que comprovar que les opcions que necessitem del webservice estan activades:

Home->Site administration->Plugins->Web services->External services->Functions

De moment necessitarem les següents funcions, així que si no estan activades, les activem:
core_users_get_users_by_id - per agafar la informació dels alumnes
core_user_create_users - per a crear usuaris
core_user_update_users - per modificar usuaris
core_user_delete_users - per a eliminar usuaris
core_course_get_courses - per agafar la informació dels cursos
enrol_manual_enrol_users - per matricular usuaris als cursos

¿Quina comanda es Curl?

Curl és una llibreria de funcions per a connectar amb servidor per a treballar amb ells. Es realitza amb format URL, es a dir, serveix per a realitzar accions sobre arxius que hi ha en URLs d'Internet, suportant protocols comuns com http, ftp, https, etc.
La forma mes simple d'aquesta comanda seria:

```
curl https://domain.com
```

Bàsicament mostraría el contingut d'aquesta pàgina al nostre terminal.
La comanda que nosaltres utilitzarem és:

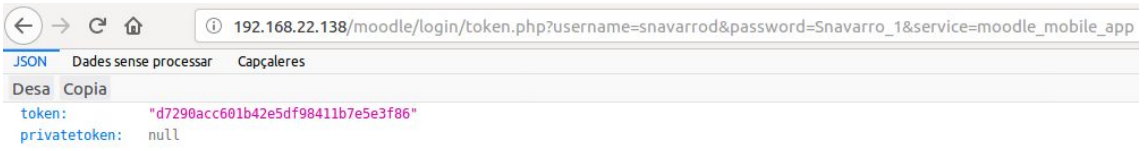
```
curl -d "..."
```

El paràmetre **-d** de curl ve de: **<data>**, el que obtindrem d'aquesta comanda es un **"post string"** del contingut que nosaltres li hem passat com a variable, com podrem veure a les captures següents.

Utilització del WebService i la comanda curl -d

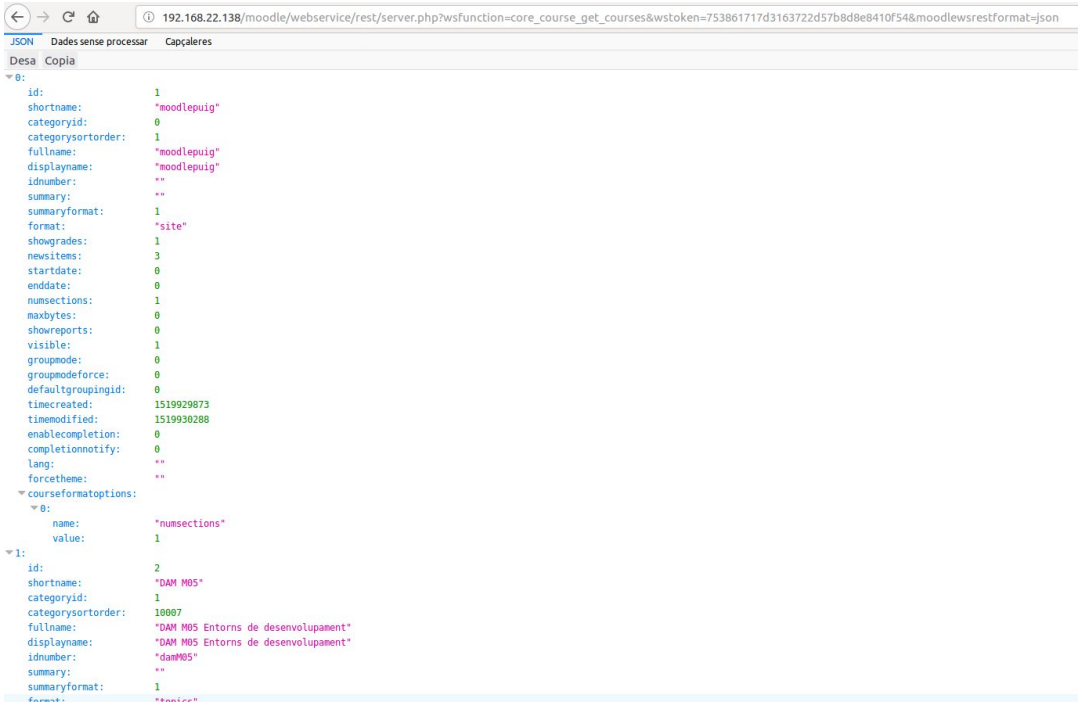
Primer hem hagut de trobar el token d'un alumne utilitzant el seu nom d'usuari i la contrasenya. Ho hem realitzat a través del webservice:

```
http://192.168.22.138/moodle/login/token.php?username=snavarrod&password=Snavarro_1&service=moodle_mobile_app
```



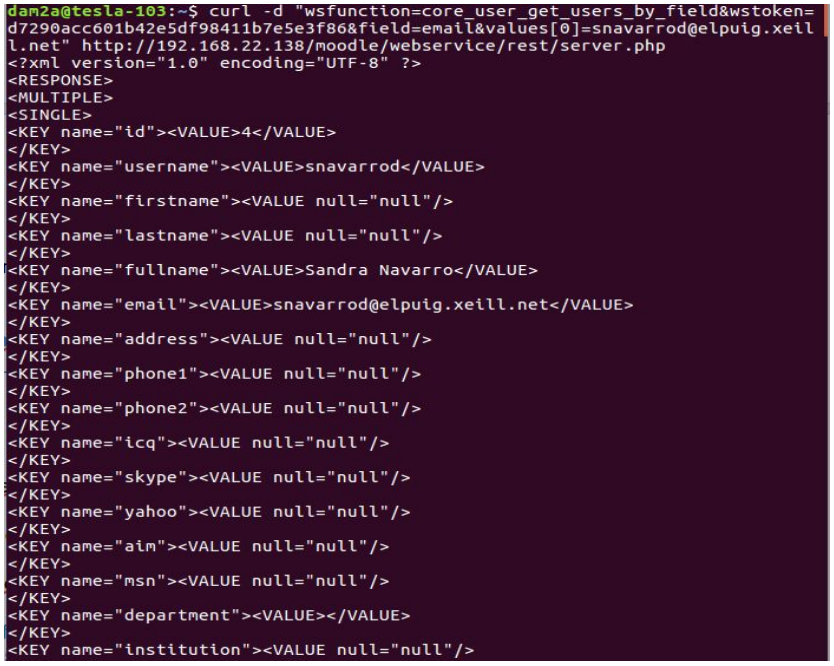
També hem buscat tots els cursos dels que disposa el nostre Moodle a través del token de l'usuari Admin, ho hem fet de dues maneres diferents, la primera es a través del webservice:

```
http://192.168.22.138/moodle/webservice/rest/server.php?wsfunction=core_course_get_courses&wstoken=753861717d3163722d57b8d8e8410f54&moodlewsrestformat=json
```



I la segona a partir d'una comanda al Terminal:

```
curl -d "wsfunction=core_course_get_courses&wstoken=753861717d3163722d57b8d8e8410f54"http://192.168.22.138/moodle/webservice/rest/server.php
```

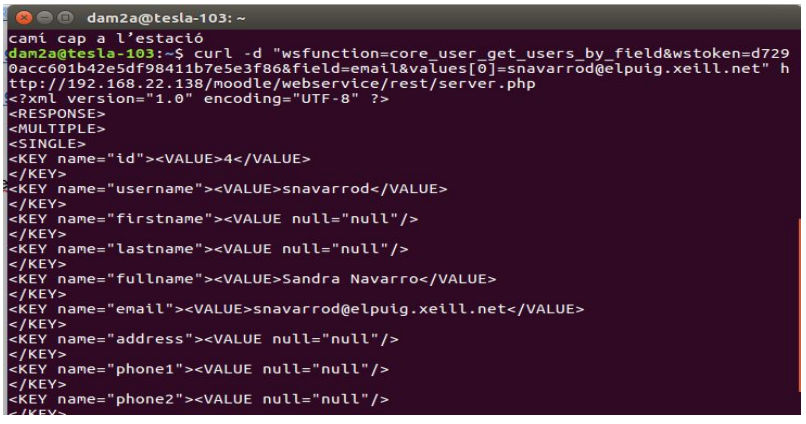


Per obtenir la ID dels usuaris hem realitzat el següent:

(core_user_get_users_by_id serveix per obtenir la informació dels usuaris)

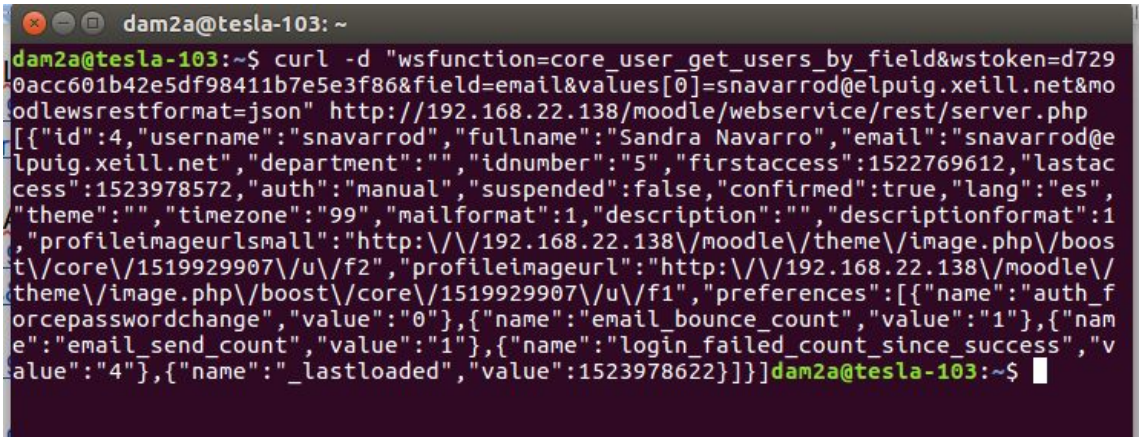
- Sortida en format XML:

```
curl -d
"wsfunction=core_user_get_users_by_field&wstoken=d7290acc601b42e5df98411b7e
5e3f86&field=email&values[0]=snavarrod@elpuig.xeill.net"
http://192.168.22.138/moodle/webservice/rest/server.php
```



- Sortida en format JSON:

```
curl -d
"wsfunction=core_user_get_users_by_field&wstoken=d7290acc601b42e5df98411b7e
5e3f86&field=email&values[0]=snavarrod@elpuig.xeill.net&moodlewsrestformat=
json" http://192.168.22.138/moodle/webservice/rest/server.php
```



La diferencia bàsica entre les dues sortides des del punt de vista de la comanda en sí, es simplement afegint-hi al final de la comanda *&moodlewsrestformat=json*.

Obtenim els cursos d'un usuari utilitzant la seva ID, ho hem fet de dues maneres diferents, la primera es a través del webservice:

```
http://192.168.22.138/moodle/webservice/rest/server.php?wsfunction=core_enr
ol_get_users_courses&wstoken=d7290acc601b42e5df98411b7e5e3f86&userid=4
```

I des del terminal:

```
curl -d
"wsfunction=core_enrol_get_users_courses&wstoken=d7290acc601b42e5df98411b7e
5e3f86&userid=4" http://192.168.22.138/moodle/webservice/rest/server.php
```

```
dam2a@tesla-103:~$ curl -d "wsfunction=core_enrol_get_users_courses&wstoken=d7290acc601b42e5df98411b7e5e3f86
&userid=4" http://192.168.22.138/moodle/webservice/rest/server.php
<?xml version="1.0" encoding="UTF-8" ?>
<RESPONSE>
<MULTIPLE>
<SINGLE>
<KEY name="id"><VALUE>8</VALUE>
</KEY>
<KEY name="shortname"><VALUE>DAM M03</VALUE>
</KEY>
<KEY name="fullname"><VALUE>DAM M03 Programació Orientada a Objectes</VALUE>
</KEY>
<KEY name="enrolledusercount"><VALUE>4</VALUE>
</KEY>
<KEY name="idnumber"><VALUE>damM03</VALUE>
</KEY>
<KEY name="visible"><VALUE>1</VALUE>
</KEY>
<KEY name="summary"><VALUE></VALUE>
</KEY>
<KEY name="summaryformat"><VALUE>1</VALUE>
</KEY>
<KEY name="format"><VALUE>topics</VALUE>
</KEY>
<KEY name="showgrades"><VALUE>1</VALUE>
</KEY>
<KEY name="lang"><VALUE></VALUE>
</KEY>
<KEY name="enablecompletion"><VALUE>1</VALUE>
</KEY>
<KEY name="category"><VALUE>1</VALUE>
```

Per trobar les tasques d'un usuari amb la seva ID i el seu token (tant les pendents d'entregar i les ja entregades), ho hem fet de dues maneres diferents, la primera es a través del webservice:

```
http://192.168.22.138/moodle/webservice/rest/server.php?wsfunction=mod_assi
gn_get_assignments&wstoken=d7290acc601b42e5df98411b7e5e3f86&courseids[0]=8
```

I des del terminal:

```
curl -d
"wsfunction=mod_assign_get_assignments&wstoken=d7290acc601b42e5df98411b7e5e
3f86&courseids[0]=8"
http://192.168.22.138/moodle/webservice/rest/server.php
```

```
dam2a@tesla-103:~$ curl -d "wsfunction=mod_assign_get_assignments&wstoken=d7290acc601b42e5df98411b7e5e3f86&c
ourseids[0]=8"  http://192.168.22.138/moodle/webservice/rest/server.php
<?xml version="1.0" encoding="UTF-8" ?>
<RESPONSE>
<SINGLE>
<KEY name="courses"><MULTIPLE>
<SINGLE>
<KEY name="id"><VALUE>8</VALUE>
</KEY>
<KEY name="fullname"><VALUE>DAM M03 Programació Orientada a Objectes</VALUE>
</KEY>
<KEY name="shortname"><VALUE>DAM M03</VALUE>
</KEY>
<KEY name="timemodified"><VALUE>1520440306</VALUE>
</KEY>
<KEY name="assignments"><MULTIPLE>
<SINGLE>
<KEY name="id"><VALUE>62</VALUE>
</KEY>
<KEY name="cmid"><VALUE>62</VALUE>
</KEY>
<KEY name="course"><VALUE>8</VALUE>
</KEY>
<KEY name="name"><VALUE>Exercicis d'introducció al java i a la POO</VALUE>
</KEY>
<KEY name="nosubmissions"><VALUE>0</VALUE>
</KEY>
<KEY name="submissiondrafts"><VALUE>0</VALUE>
</KEY>
<KEY name="sendnotifications"><VALUE>0</VALUE>
</KEY>
<KEY name="sendlatenotifications"><VALUE>0</VALUE>
</KEY>
<KEY name="sendstudentnotifications"><VALUE>1</VALUE>
</KEY>
<KEY name="duedate"><VALUE>1521500400</VALUE>
</KEY>
<KEY name="allowsubmissionsfromdate"><VALUE>1520895600</VALUE>
</KEY>
<KEY name="grade"><VALUE>100</VALUE>
```

5. Conclusions

La conclusió que hem tret d'aquest projecte és que és fàcil aprendre a utilitzar eines noves pel nostre compte si s'investiga correctament, tenint les idees clares del que es vol. El fer el projecte amb Java que és un idioma de programació que hem estudiat al llarg del curs ens ha ajudat molt ja que teníem uns coneixements bàsics de programació. Per altra banda, se'ns ha fet complicat entendre com funcionava la API del Moodle, ja que no havíem treballat mai amb ell, i no sabíem com utilitzar la informació que ens proporcionava Moodle.

Per compartir el codi entre els membres del projecte vam utilitzar l'aplicació GitHub. Encara que és una manera útil de compartir projectes, vam tenir problemes a l'hora d'actualitzar el codi ja que a vegades funcionava correctament a uns ordinadors i fallava en altres, o no s'actualitzava bé.

Hem complert tots els objectius proposats del projecte, encara que el canvi d'idioma de l'aplicació ha sigut el més complicat. Hem aconseguit traduir els missatges del nostre programa però Telegram no té cap manera de traduir les comandes que se li proporcionen al bot.

Hem seguit la planificació correctament, però el imprevist de l'idioma ens ha portat més temps de l'esperat, retrasant un parell de dies el treball de redactat de la memòria i la presentació.
El desenvolupament seguit durant el projecte ha variat una mica de l'inicialment proposat, ja que algunes parts del disseny les hem anat implementant mentre comprovabem el correcte funcionament del codi.

En el futur ens agradaria que els alumnes que arribin a segon de DAM trobin aquest projecte interessant i acabin de millorar-lo, implementant els cursos que ara mateix no estàn.

6. Glossari

- API: Application Programming Interface.
- JSON: JavaScript Object Notation.
- SQL: Structured Query Language.
- WebService: Servei web.
- HTTPS: Hypertext Transfer Protocol Secure.
- Bot de Telegram: Aplicacions de tercers que s'executen a l'interior de Telegram. Els usuaris poden interactuar amb bots enviant missatges, ordres i peticions en línia. Aquests es controlen mitjançant sol·licituds HTTPS a la API bot.

7. Bibliografia

Web: Documentació oficial de Telegram
- https://core.telegram.org/bots
- https://core.telegram.org/bots#6-botfather

Web: Altres projectes amb bots de Telegram
- https://monsterdeveloper.gitbooks.io/writing-telegram-bots-on-java/content/chapter1.html
- https://github.com/MonsterDeveloper/java-telegram-bot-tutorial
- https://github.com/LuisGi93/pradobot/tree/master/test
- https://github.com/rubenlagus/TelegramBots
- https://github.com/Eng-Fouad/JTelegramBot
- https://github.com/leocus/telegramBotUtilities

Web: Moodle
- https://moodle.org/
- https://docs.moodle.org/33/en/Step-by-step_Installation_Guide_for_Ubuntu

Web: Base de dades SQLite
- http://www.sqlitetutorial.net/sqlite-java/sqlite-jdbc-driver/
- https://github.com/gerardfp/SimpleSQLite/tree/master/src/com/company

8. Annexos

MANUAL D'USUARI

Què és el bot de Telegram del Puig?

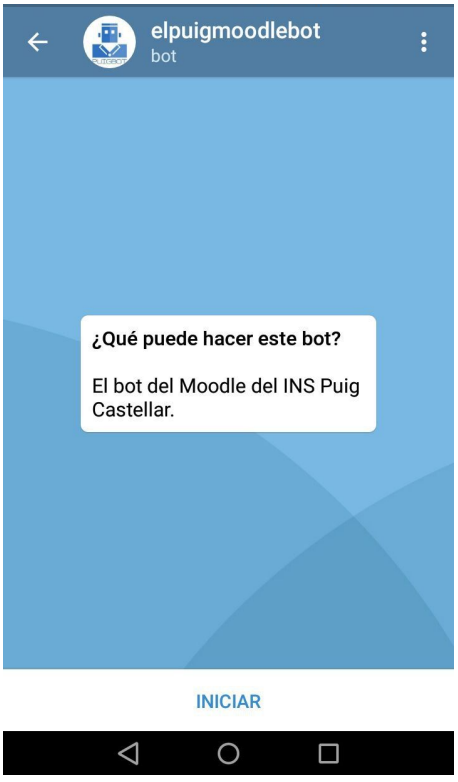
Els bots són aplicacions de tercers que s'executen a l'interior de Telegram. Els usuaris poden interactuar amb bots enviant missatges, ordres i peticions en línia. Aquests es controlen mitjançant sol·licituds HTTPS a la API bot.

El bot del Moodle del Puig serveix per que els usuaris puguin fer consultes relatives a la seva informació emmagatzemada al Moodle. Amb ordres sencilles els usuaris poden consultar horaris, entregues, exàmens i fins i tot les seves notes finals.

Començar

Per accedir al bot l'usuari ha de consultar el cercador de Telegram. Tant si escriu elpuigmoodlebot com si escriu elmoodlepuigbot apareixerà al llistat.

Per començar a utilitzar el bot l'usuari haurà de prémer el botó "Iniciar". El bot donarà la benvinguda.



Iniciar sessió

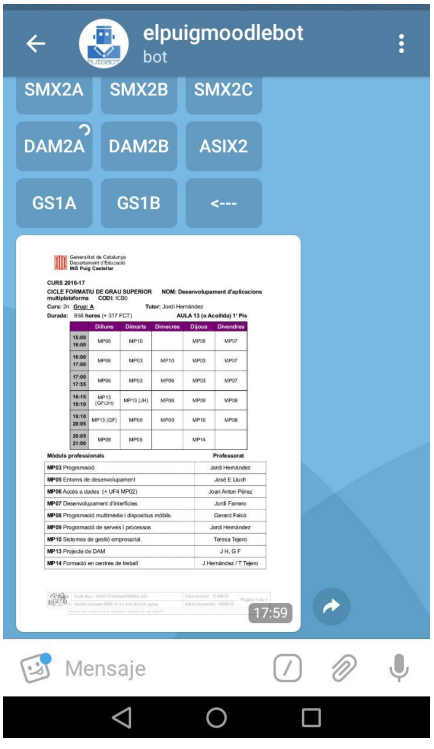
Per iniciar sessió es necessari escriure la comanda /login seguida del nom d'usuari de Moodle i la contrasenya. Si el format d'entrada no és correcte el bot retornarà un missatge d'ajuda per a l'usuari.

Consultar horaris

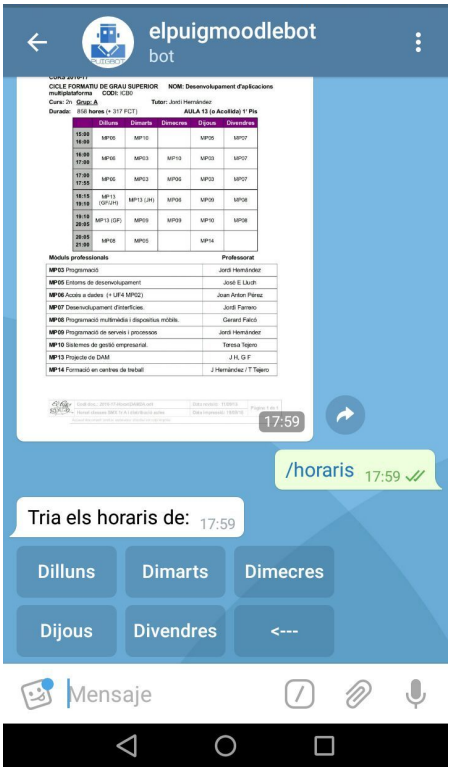
Per accedir als horaris la comanda és /horaris. Aquesta retorna un parell de botons amb els grups i els professors. Per els grups ens retornarà una serie de botons amb els grups disponibles. En fer clic en l'horari desitjar el bot retornarà una foto del horari.

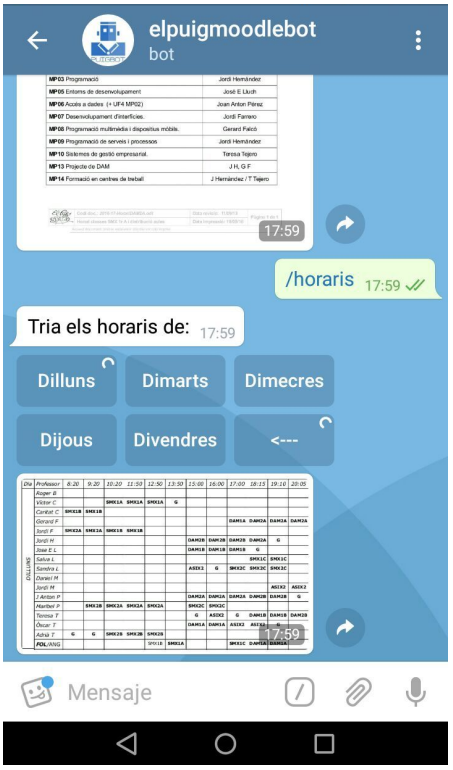

elpuigmoodlebot
bot
¿Qué puede hacer este bot?
El bot del Moodle del INS Puig Castellar.
28 de mayo
/start 17:58
Benvingut Ama
Aquest és el bot d'El Puig 17:58
/login amartinez Amartinez_1 17:58
Login realitzat amb èxit 17:58
/horaris 17:58
Tria de qui vols veure horarisssssss 17:58
Grups
Professors
Mensaje


elpuigmoodlebot
bot
28 de mayo
/start 17:58
Benvingut Ama
Aquest és el bot d'El Puig 17:58
/login amartinez Amartinez_1 17:58
Login realitzat amb èxit 17:58
/horaris 17:58
Tria el grup: 17:58
SMX1A
SMX1B
SMX1C
SMX2A
SMX2B
SMX2C
DAM2A
DAM2B
ASIX2
GS1A
GS1B
<---
Mensaje


elpuigmoodlebot
bot
SMX2A
SMX2B
SMX2C
DAM2A
DAM2B
ASIX2
GS1A
GS1B
<---
17:59
Mensaje

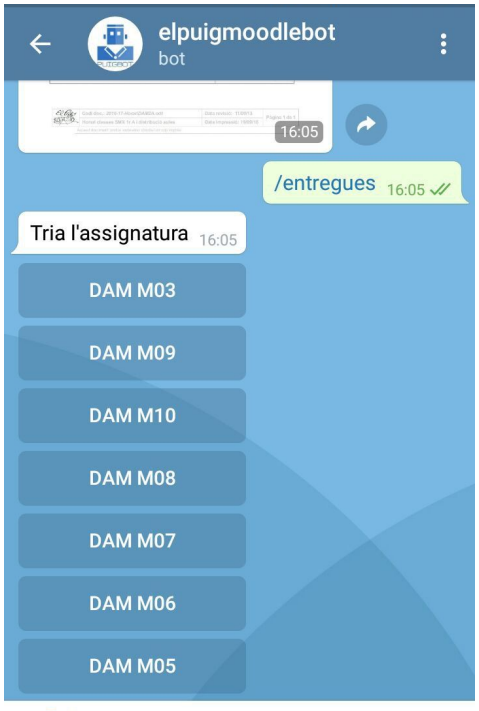
Per els horaris dels professors, apareixen els dies de la setmana.





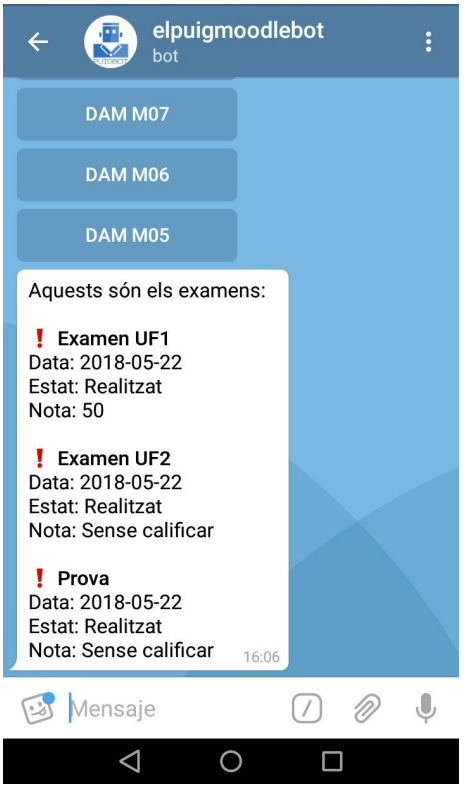
Consultar entregues

Per consultar les entregues s'utilitza la comanda /entregues. Aquesta retorna un llistat d'assignatures. L'usuari ha d'escollir una. En fer clic al botó de l'assignatura que vol, sortirà un llistat de les entregues.



Consultar exàmens

Per consultar les entregues s'utilitza la comanda /examens. Aquesta retorna un llistat d'assignatures. L'usuari ha d'escollir una. En fer clic al botó de l'assignatura que vol, sortirà un llistat dels exàmens.



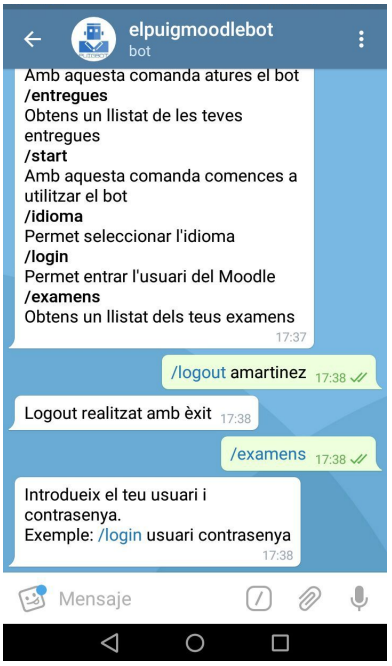
Nota per a professors: És necessari que els examens tinguin les paraules "examen" o "prova" a Moodle, sino apareixeran com entregues i no exàmens.

Consultar notes finals

Per consultar les entregues s'utilitza la comanda /notes. Aquesta retorna un llistat amb les notes dels trimestres i els enllaços al butlletí *.pdf*.

Tancar sessió

Per tancar sessió es necessari escriure la comanda /login seguida del nom d'usuari. Si el format d'entrada o l'usuari intenta accedir a les dades sense estar *loggeat*, el bot retornarà un missatge d'ajuda per a l'usuari.



Canviar idioma

El bot també té una opció per canviar l'idioma. Això es fa amb la comanda /idiomes. Apareixerà un menú on l'usuari pot seleccionar el llenguatge que prefereix.