**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

**(Universidad del Perú, DECANA DE AMÉRICA)**

**Facultad Ingeniería de Sistemas e Informática**

**E.A.P. de Ingeniería de Sistemas**



BASE DE DATOS II

TEMA: SISTEMA DE GESTION DE PACIENTES EN UNA CLINICA

GRUPO N.º 9

Docente:

Sotelo Bedón, Marcos

Alumnos:

* + Gutiérrez Jorgechagua, Juan 14200137
  + García Agüero, Oscar 16200175

27 de febrero de 2020

Ciudad Universitaria, Lima, Perú

ÍNDICE

[I. PRESENTACION DEL CASO 2](#_Toc33646756)

[II. MODELADO UML- MER 0](#_Toc33646757)

[III. MAPEO UML - RELACIONAL 0](#_Toc33646758)

[IV. POBLADO DE DATOS 0](#_Toc33646759)

[V. QUERIES CON GRADO DE DIFICULTAD INCREMENTAL 2](#_Toc33646760)

[VI. TABLA CRITICA CON SP 3](#_Toc33646761)

[VII. SP ORIENTADO A TOMA DE DECISIONES 4](#_Toc33646762)

[VIII. TRIGGER PARA ATRIBUTO REDUNDANTE 5](#_Toc33646763)

[IX. TRANSACCIONES 6](#_Toc33646764)

[X. CONTROL DE CONCURRENCIA DE TRANSACCIONES 7](#_Toc33646765)

[XI. RECUPERACIÓN DE TRANSACCIONES 8](#_Toc33646766)

[XII. APLICACIÓN (PROTOTIPO) QUE MUESTRE LA INTEGRACIÓN CON LA BASE DE DATOS 9](#_Toc33646767)

[XIII. ARQUITECTURA DE SEGURIDAD DE BASE DE DATOS 11](#_Toc33646768)

[XIV. CONTROL DE ACCESO A LA BASE DE DATOS 12](#_Toc33646769)

[XV. OPTIMIZACIÓN DE CONSULTAS 14](#_Toc33646770)

[XVI. MEJORA DE RENDIMIENTO A PARTIR DEL AFINAMIENTO (TUNNING) 16](#_Toc33646771)

[XVII. BIBLIOGRAFÍA 17](#_Toc33646772)

# PRESENTACION DEL CASO

Las tecnologías de información y comunicación (TIC), junto con las metodologías avanzadas de procesamiento de la información han evolucionado significativamente, influyendo activamente en nuestra sociedad. En salud este impacto estaba orientado a la optimización de procesos, al incremento de productividad y la gestión financiera, en lo que conocemos como la era de los “sistemas de información hospitalarios”. Sin embargo, al entender el flujo de trabajo y contexto clínico como una serie de procesos complejos en los que el paciente y su interacción con el sistema de salud son el centro de la escena, estos sistemas evolucionaron hacia “sistemas de información en salud” donde la captura, intercambio, almacenamiento, acceso y gestión de información clínica adquieren un rol fundamental, con el objetivo de contribuir a la calidad y eficiencia de la atención de salud, mejorar la accesibilidad a sus servicios y el conocimiento médico.

En este contexto, la Clínica Mundo Salud requiere implementar un nuevo sistema que le ayude gestionar la información de los pacientes, las citas, atenciones, intervenciones e internamientos, así como la gestión de sus medicamentos en farmacia.

En este proyecto pondremos más énfasis en la base de datos que almacenara la información producida por el sistema de gestión de pacientes.

# MODELADO UML- MER

Imagen que contiene texto, mapa

Descripción generada automáticamente

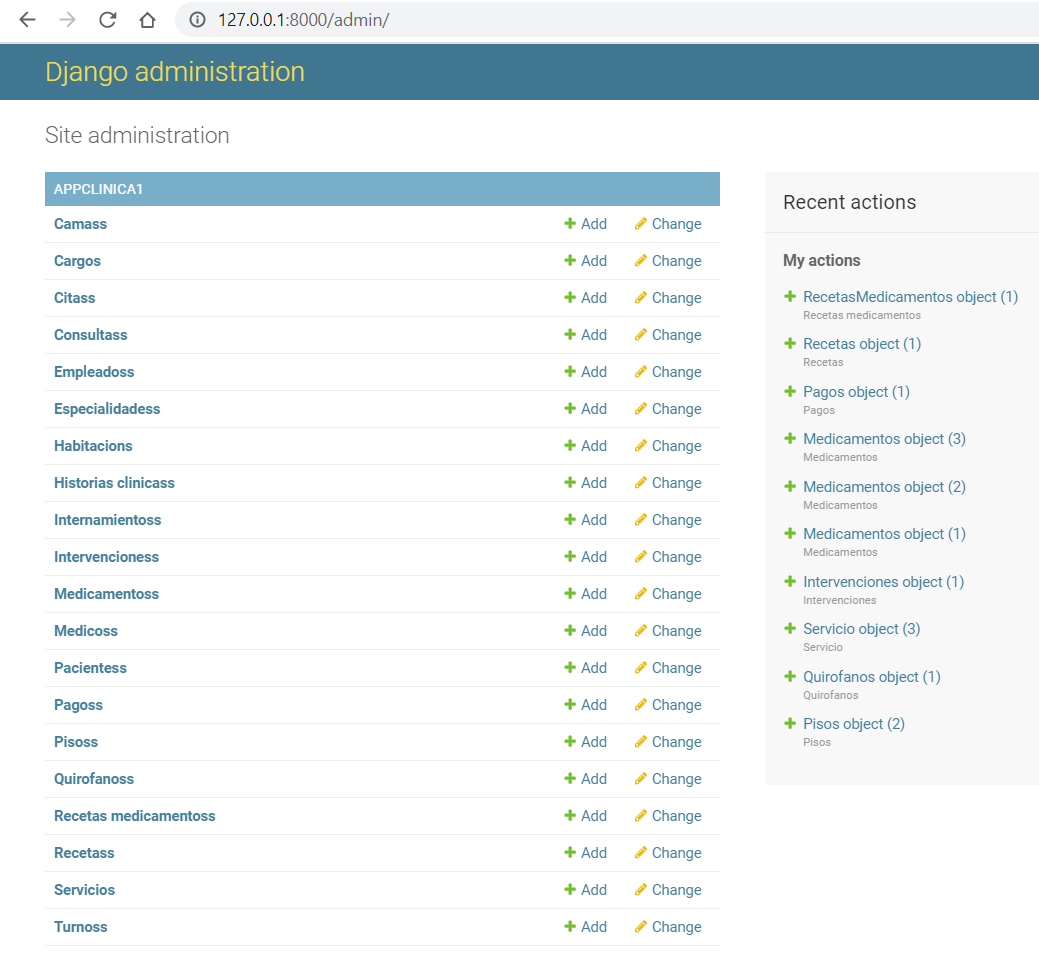
# MAPEO UML - RELACIONAL

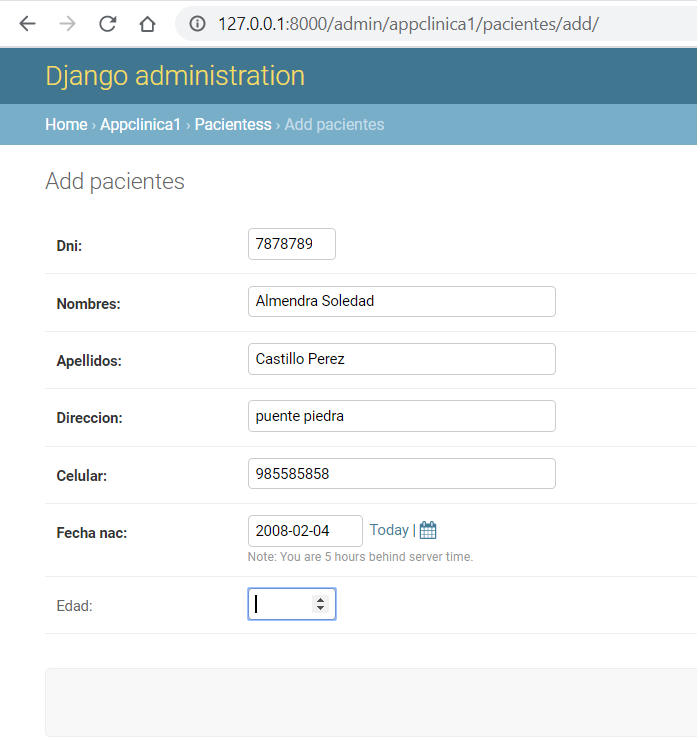
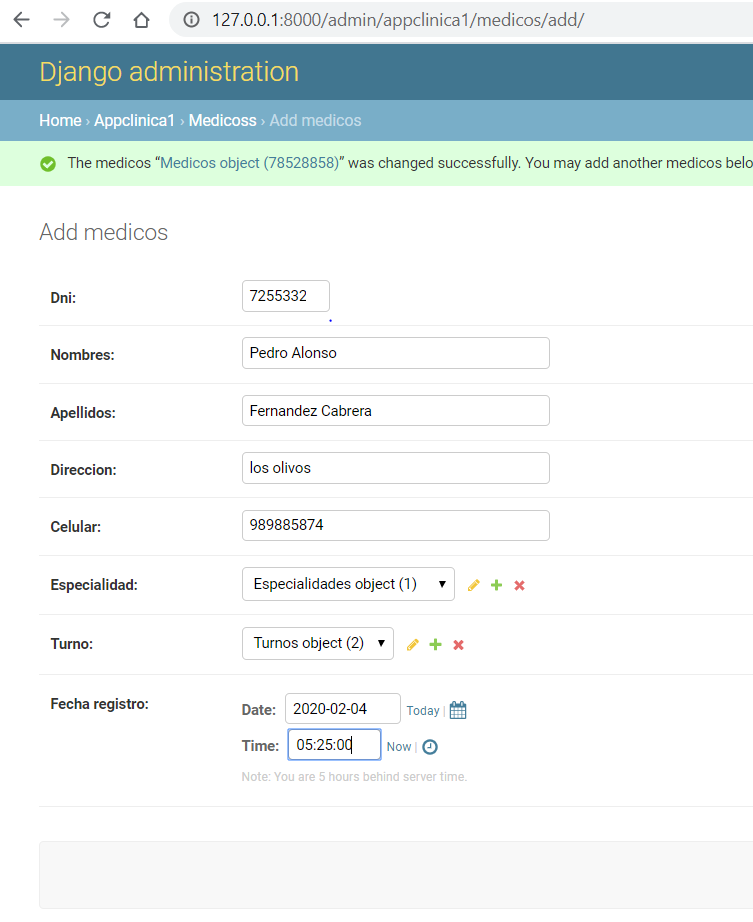
Captura de pantalla de un celular

Descripción generada automáticamente

# POBLADO DE DATOS

El poblado de datos lo realizamos por medio de la aplicación web realizada con el framework Django.





# QUERIES CON GRADO DE DIFICULTAD INCREMENTAL

select nombres , apellidos , dni from medicos;

select nombres , apellidos , dni from pacientes;

SELECT concat(m.apellidos,' ', m.nombres)as medico ,e.nombre as especialidad FROM medicos mjoin especialidades e on m.especialidad=e.id;

select DIA\_TURNO , HORA\_INICIO , HORA\_FIN , cuposfrom turnos;

SELECT concat(m.apellidos,' ', m.nombres)as MEDICO ,concat(t.DIA\_TURNO , ' de ',t.HORA\_INICIO, ' a ',t.HORA\_FIN) as HORARIOS FROM medicos mjoin turnos t on m.turno=t.id;

SELECT c.ID , p.DNI,concat(p.apellidos,' ', p.nombres)as PACIENTE,concat(m.apellidos,' ', m.nombres)as MEDICOS ,c.FECHA\_CITA as FECHA\_PROGRAMADAFROM citas cjoin pacientes p on c.PACIENTE=p.DNIjoin medicos M on c.medico=m.DNI;

select nombres , apellidos , dni from medicos;

select nombres , apellidos , dni from pacientes;

SELECT concat(m.apellidos,' ', m.nombres)as medico ,e.nombre as especialidad FROM medicos mjoin especialidades e on m.especialidad=e.id;

select DIA\_TURNO , HORA\_INICIO , HORA\_FIN , cuposfrom turnos;

SELECT concat(m.apellidos,' ', m.nombres)as MEDICO ,concat(t.DIA\_TURNO , ' de ',t.HORA\_INICIO, ' a ',t.HORA\_FIN) as HORARIOS FROM medicos mjoin turnos t on m.turno=t.id;

SELECT c.ID , p.DNI,concat(p.apellidos,' ', p.nombres)as PACIENTE,concat(m.apellidos,' ', m.nombres)as MEDICOS ,c.FECHA\_CITA as FECHA\_PROGRAMADAFROM citas cjoin pacientes p on c.PACIENTE=p.DNIjoin medicos M on c.medico=m.DNI;

# TABLA CRITICA CON SP

CREATE DEFINER=`root`@`localhost` PROCEDURE `aumentarprecio\_espec`(serid int)

BEGIN

update servicio set precio=precio+(precio\*0.5) where id=serid;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `ver\_citas`(ais varchar(30))

BEGIN

select \*from citas

where asistio like ais;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `vercamas\_vacias`(estadoc varchar(45) )

BEGIN

select c.id IDCAMA ,c.letra IDLETRA,c.estado,h.pisos\_id PISO

FROM camas c

join habitacion h on c.habitacion\_id=h.id

where c.estado like estadoc;

END

# SP ORIENTADO A TOMA DE DECISIONES

CREATE DEFINER=`root`@`localhost` PROCEDURE `aumentarprecio\_espec`(serid int)

BEGIN

update servicio set precio=precio+(precio\*0.5) where id=serid;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `ver\_citas`(ais varchar(30))

BEGIN

select \*from citas

where asistio like ais;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `vercamas\_vacias`(estadoc varchar(45) )

BEGIN

select c.id IDCAMA ,c.letra IDLETRA,c.estado,h.pisos\_id PISO

FROM camas c

join habitacion h on c.habitacion\_id=h.id

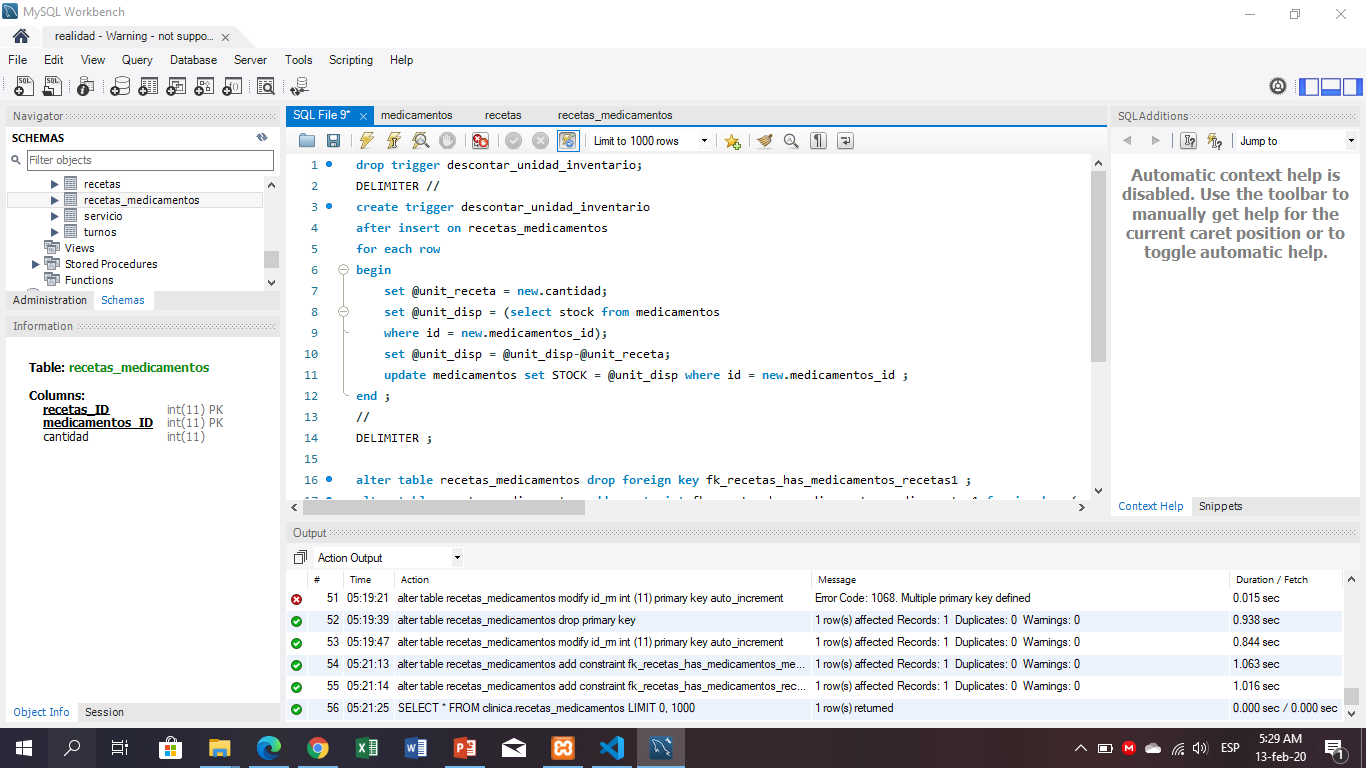
where c.estado like estadoc;

END

# TRIGGER PARA ATRIBUTO REDUNDANTE

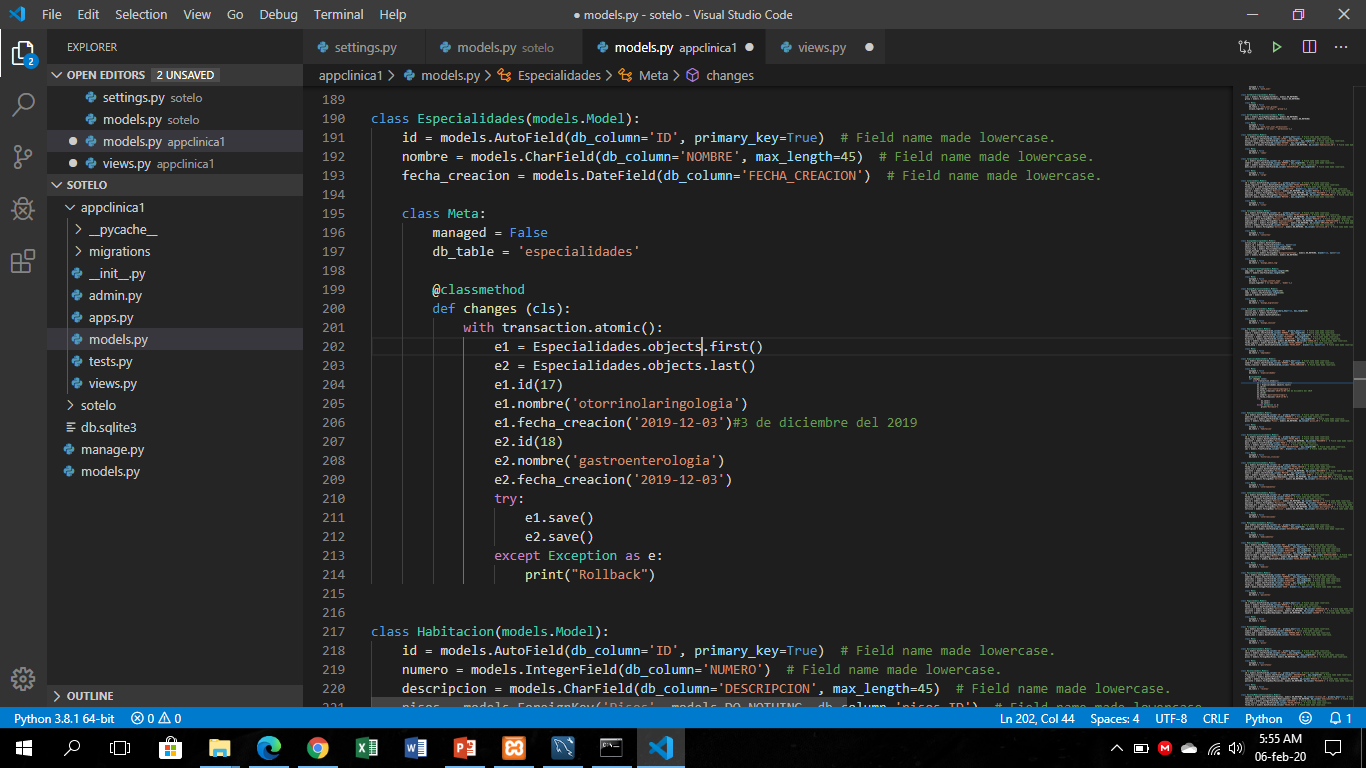
En el caso del stock de medicamentos, cuando se genera la receta es evidente que se descontará

Después de que se inserte un elemento en la tabla ‘recetas\_medicamentos’ se descontará de la tabla ‘medicamentos’



# TRANSACCIONES

Se implementará, a modo de ejemplo, en la tabla ‘especialidades’. Se usará las transacciones via modelos en django. Por ende, tanto rollback, commit y start transaction quedan implícitos gracias al uso de la librería transaction.



# CONTROL DE CONCURRENCIA DE TRANSACCIONES

* los datos que a cada rato serán agrandados

lock tables recetas\_medicamentos write, consultas write, pagos write;

* los datos que a cada rato serán consultados

lock tables medicamentos as juan read, pacientes as juan read;

# RECUPERACIÓN DE TRANSACCIONES

* Se va a respaldar todas las tablas generadas en el esquema ‘clinica’
* Para dar solución a las tablas restauradas provenientes de la secuencia django ‘python manage.py migrate’ se optará por eliminarlas
* Generación de respaldo en el disco D: (se debe colocar contraseña del servidor de base de datos)

mysqldump –u root –p clínica > D:\restaurado-clínica.sql

* Para la restauración, se recomienda guardar el archivo .sql en otra unidad de almacenamiento y luego ubicarla en el disco C:\
* Secuencia de restauracion de las tablas:

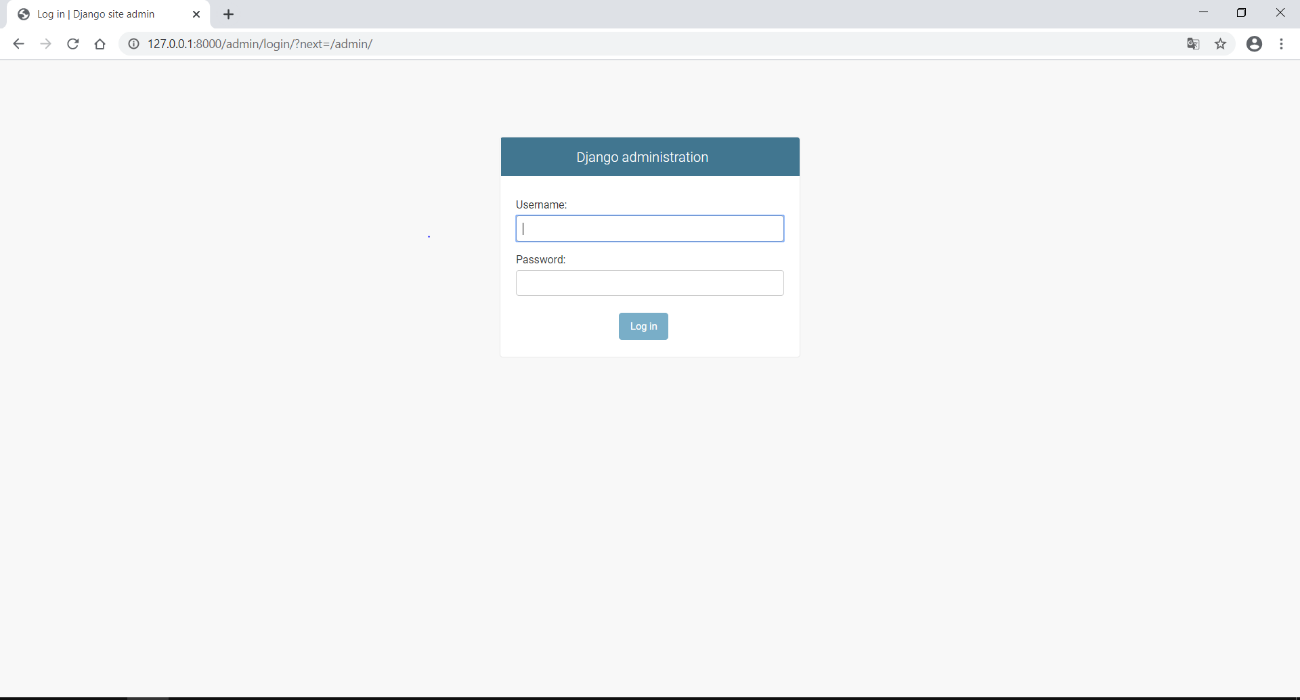
create nueva\_bd\_clinica

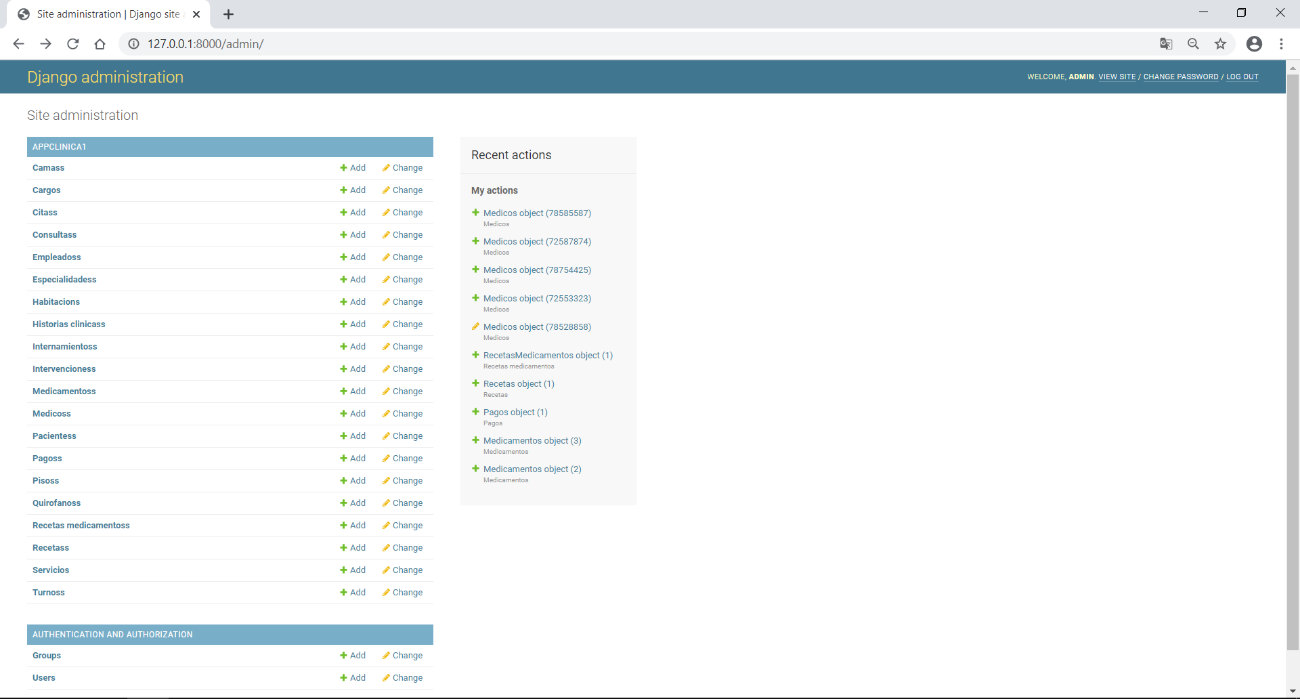
use nueva\_bd\_clinica

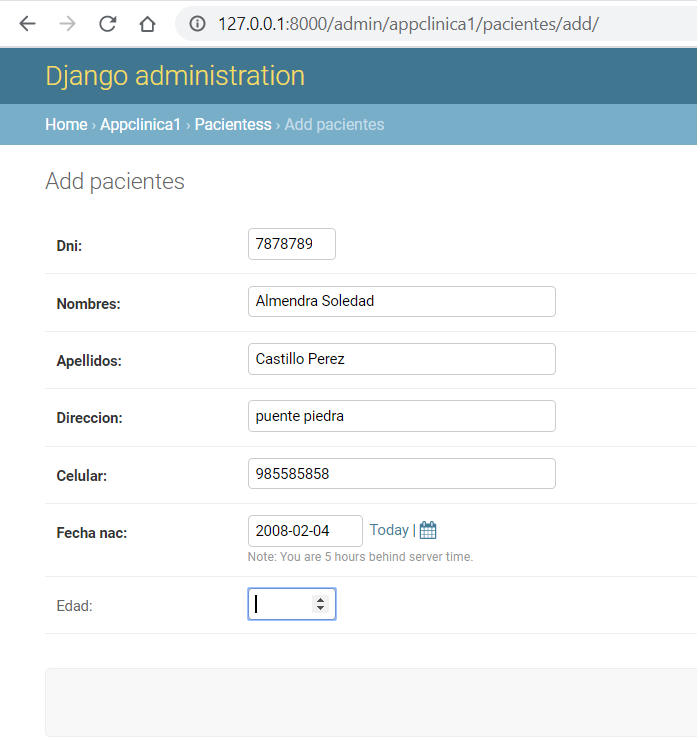
.\ C:\restaurado.clinica.sql

# APLICACIÓN (PROTOTIPO) QUE MUESTRE LA INTEGRACIÓN CON LA BASE DE DATOS

La aplicación que se realizo para el proyecto fue una aplicación web con la ayuda del framework Django.







# ARQUITECTURA DE SEGURIDAD DE BASE DE DATOS

* Control de cuentas de usuario:

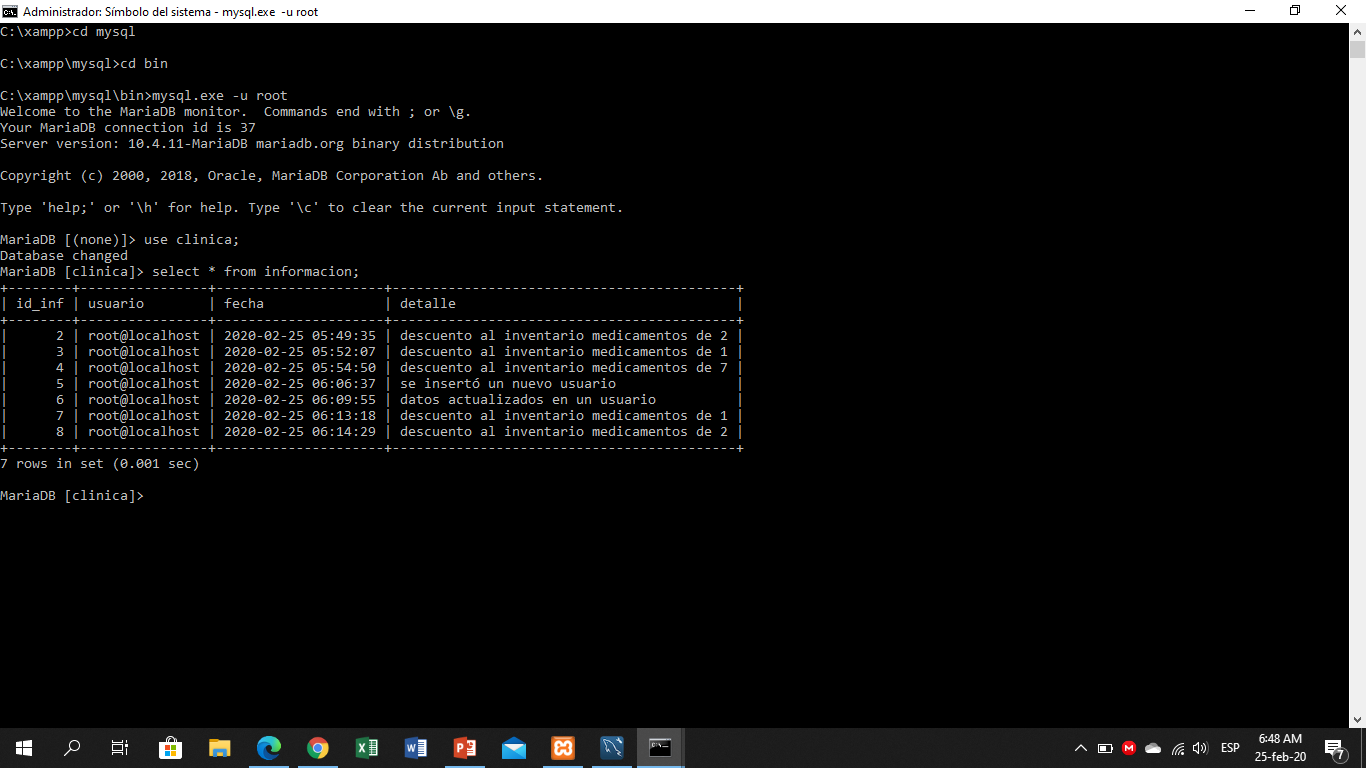
El usuario root es el único que puede crear nuevos esquemas de base de datos y usuarios

Hay otros cuatro usuarios donde solo tendrán alcance es el esquema de la clínica

* Auditoria de control:

Creación de la tabla ‘información’

Muestra los detalles sobre actualización e inserción

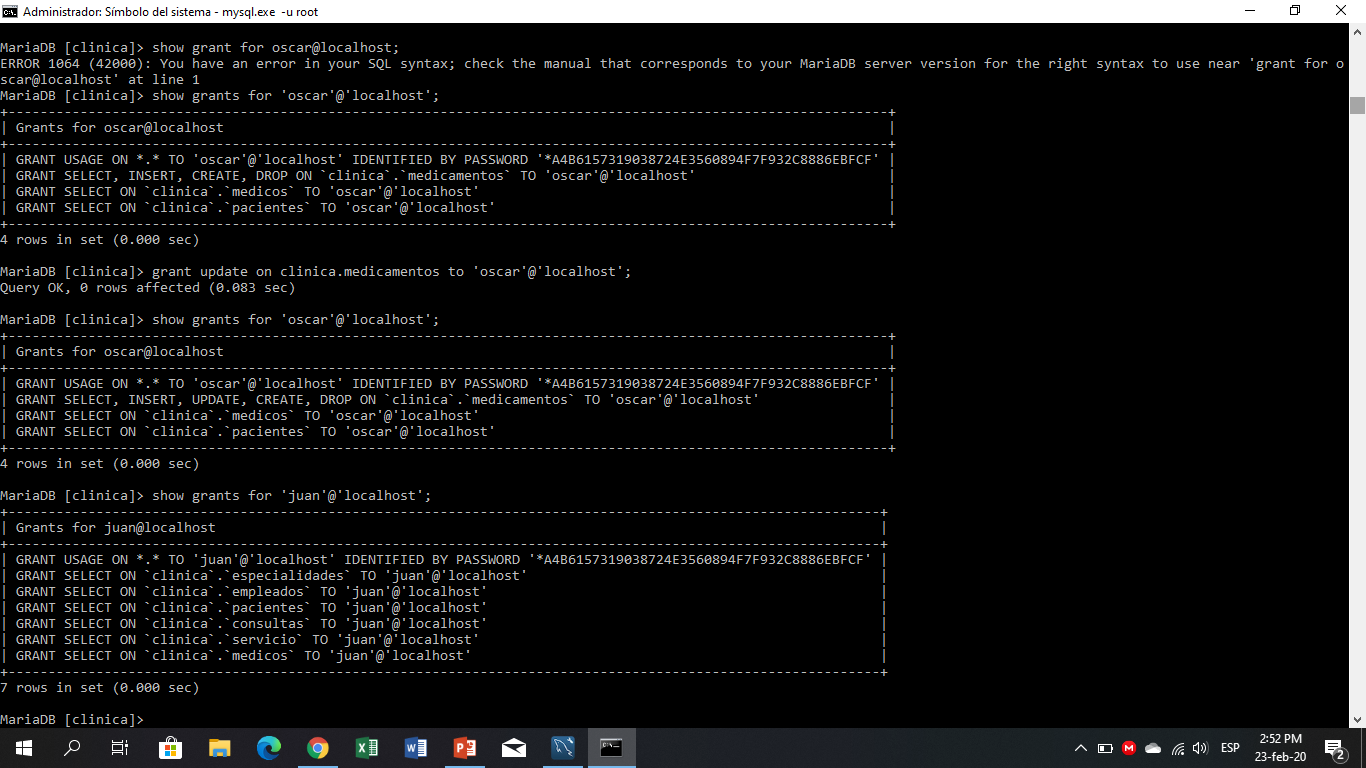


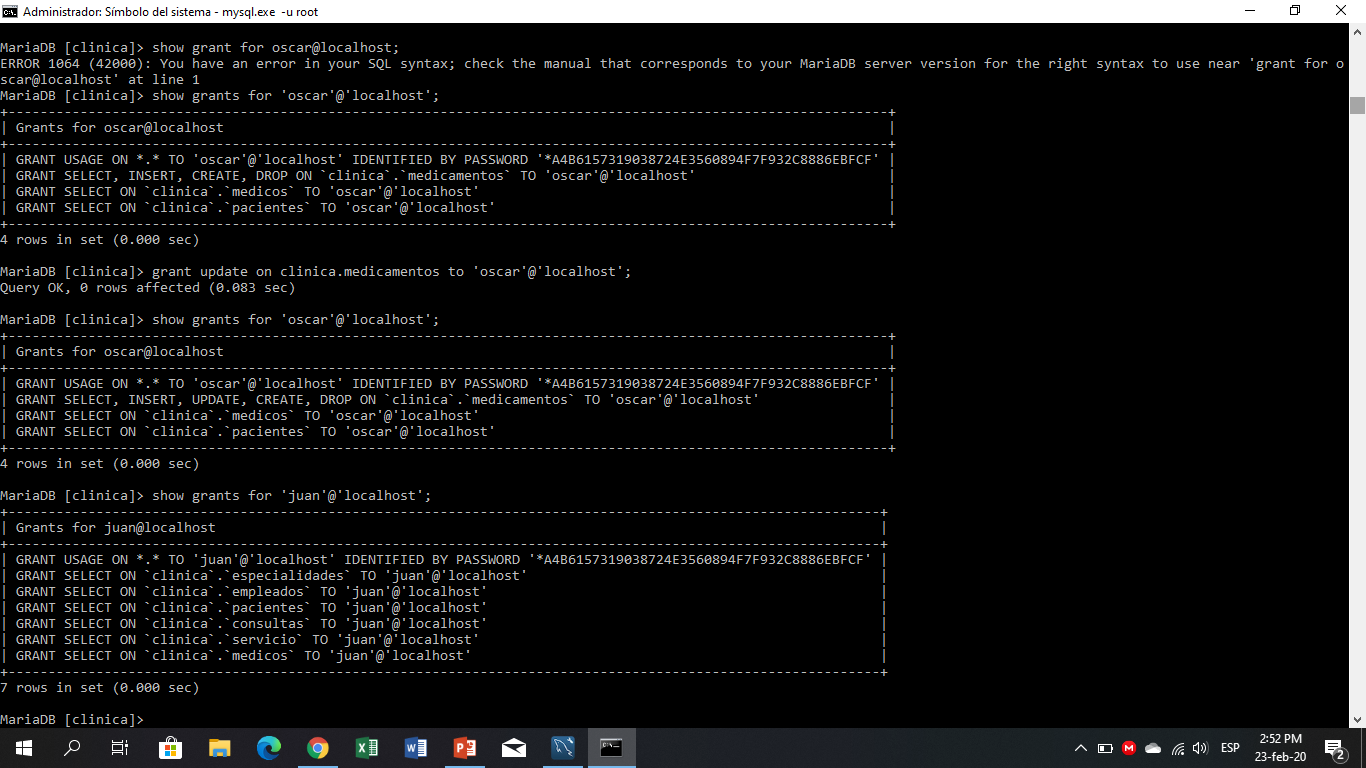
# CONTROL DE ACCESO A LA BASE DE DATOS

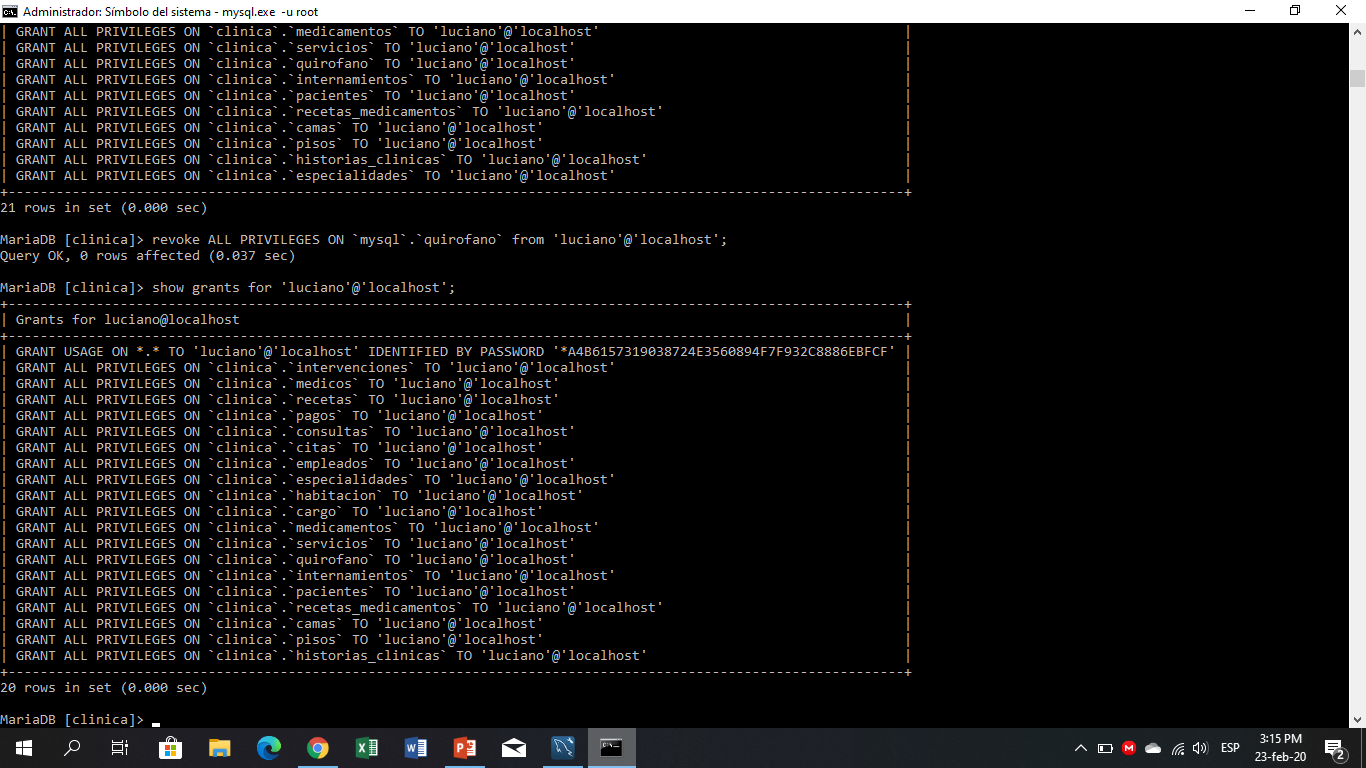
Cuatro tipos de usuario:

* Común: solo leerá las tablas correspondientes a consultas y pacientes (será encargado por juan)
* División de medicamentos: puede alterar los registros en medicamentos. Además, solo visualizará las tablas de médicos y pacientes (será encargado por Oscar)
* División de recursos: puede alterar todos los registros correspondientes al esquema de la clínica. No puede crear tablas, procedimientos o funciones (será encargado por Luciano)
* Gerente de base de datos: puede alterar la estructura del esquema de la clínica como insertar nuevas tablas, procedimientos o funciones

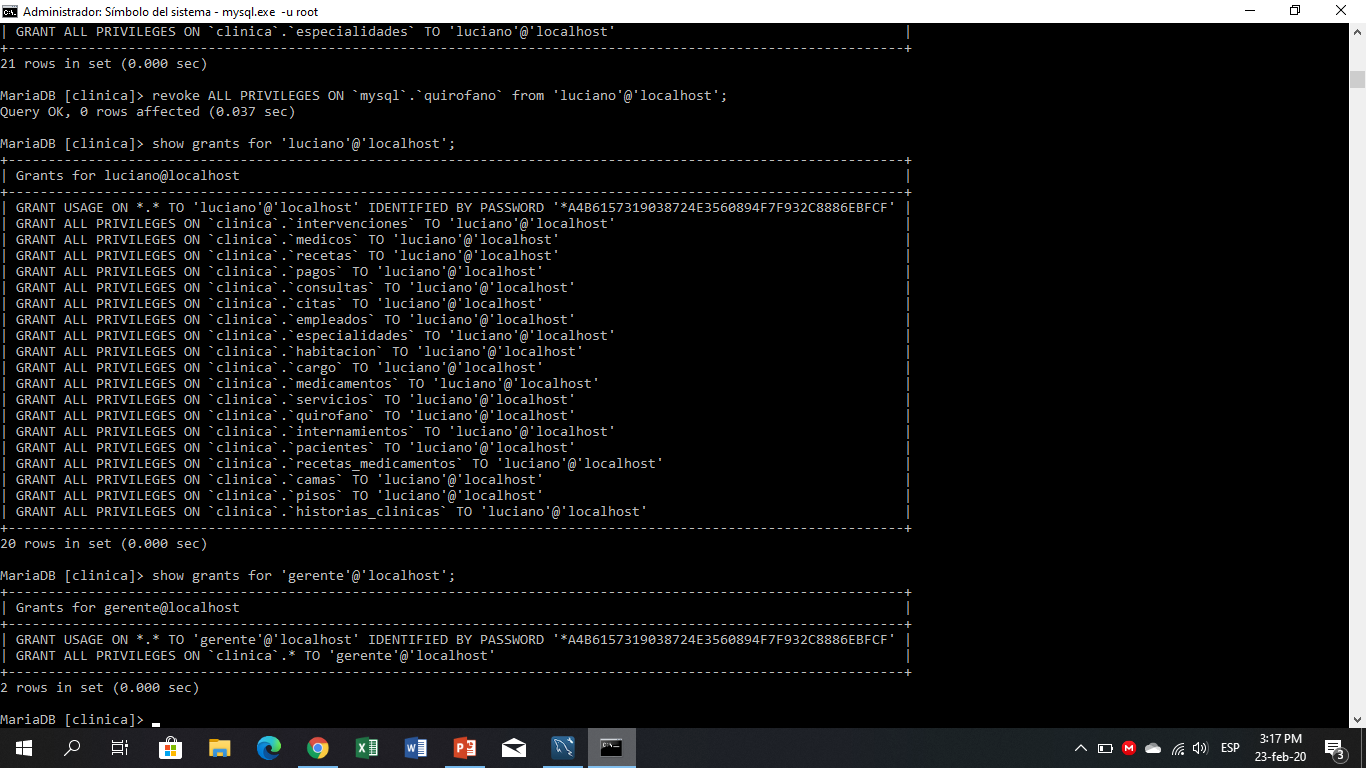
Razón: dentro de la clínica, el personal que no sea administrativo puede acceder a información limitada y entregarla a los usuarios. Personal administrativo cuenta con cuentas moderadas o de administrador.

* Usuario común (llamado Juan):

* Usuario División de medicamentos (llamado Oscar):
* Usuario División de recursos (llamado Luciano):

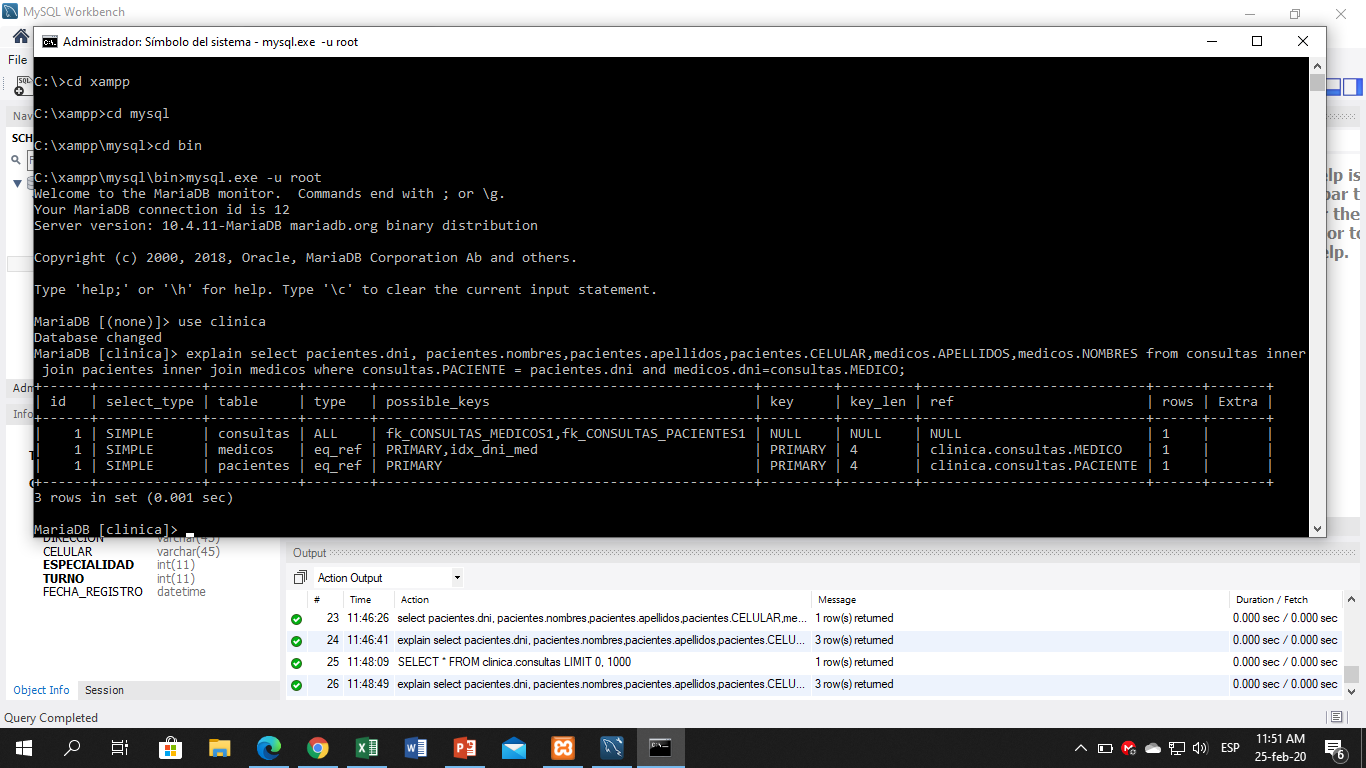


* Usuario gerente de base de datos:

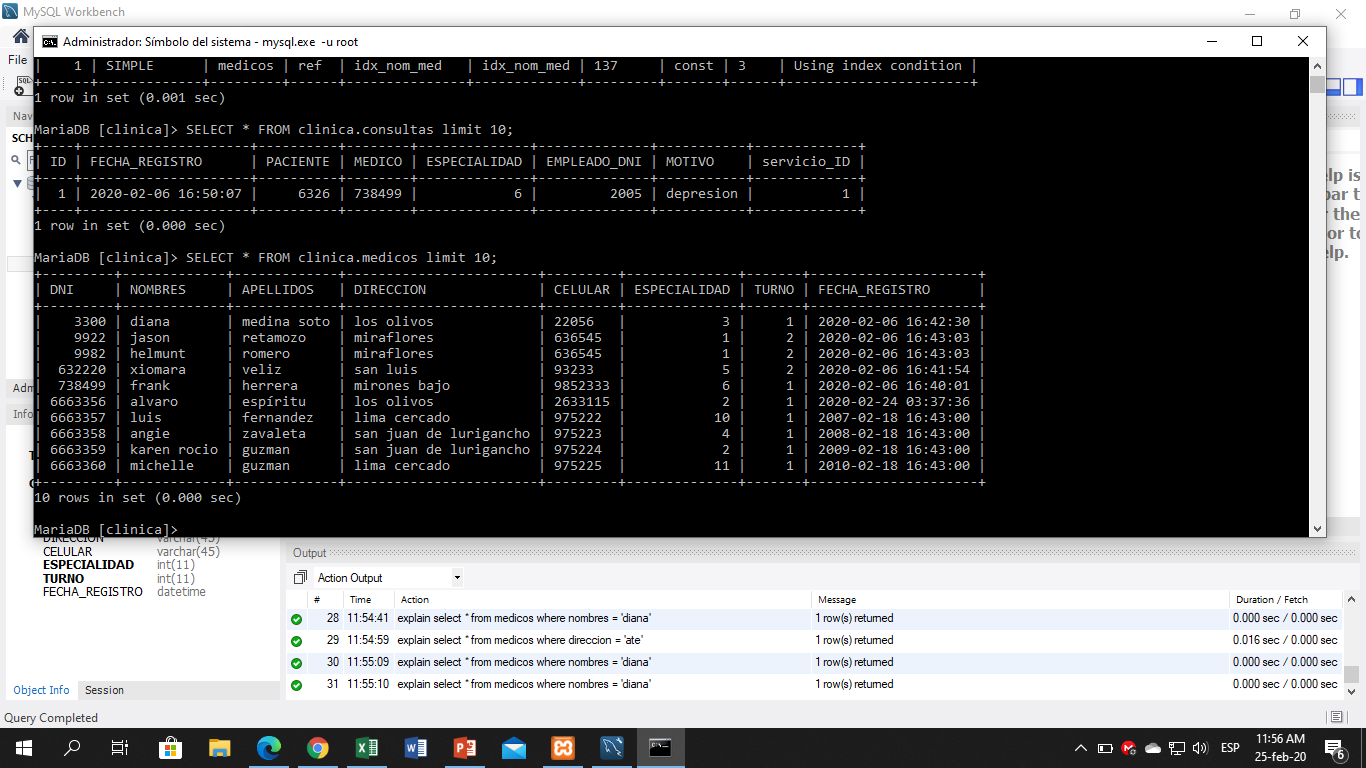


# OPTIMIZACIÓN DE CONSULTAS

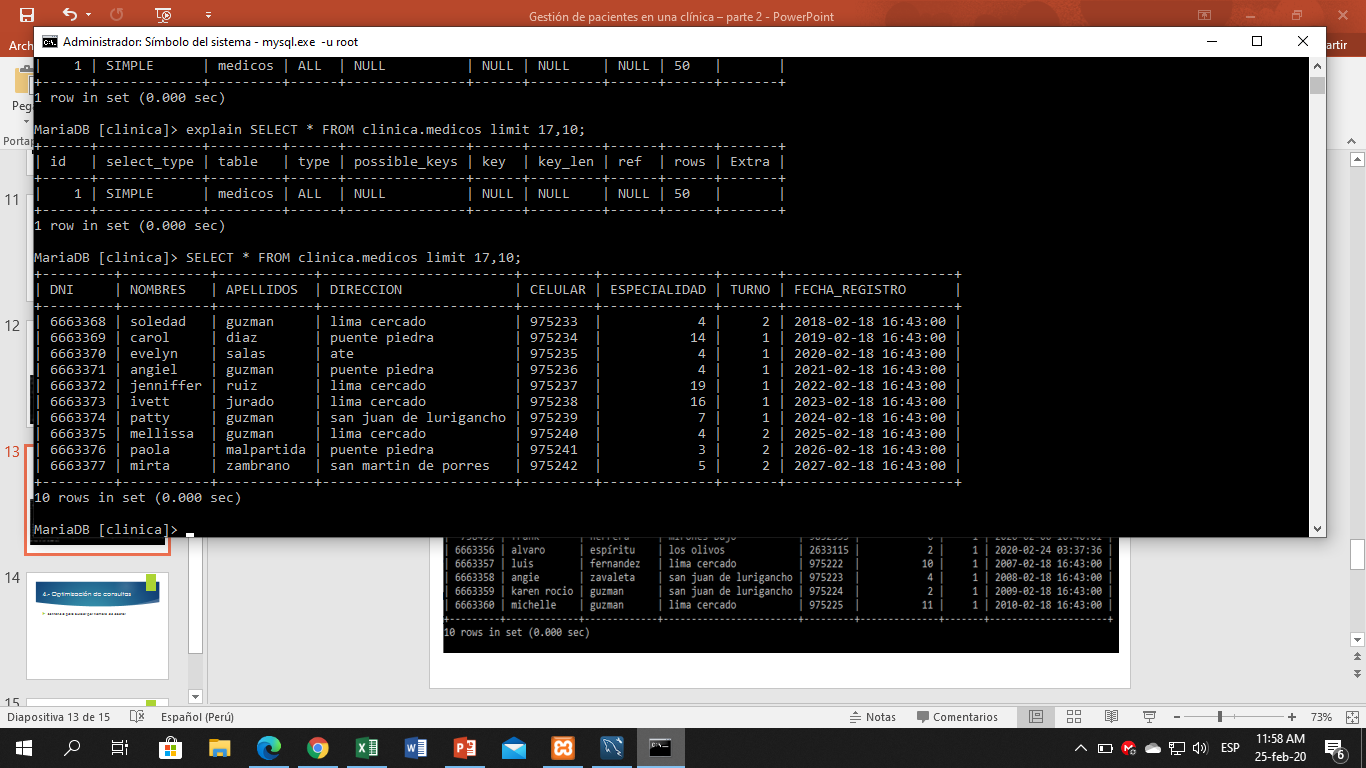
* Uso de limitadores al momento de obtener las tablas con mayor información (solo 10 columnas)
* Uso de inner joins puesto que, en la práctica, solo se necesitan los resultados compartidos entre dos a más tablas
* Uso de índices en las tablas con mayor cantidad de información
* En caso se desee mostrar ordenadamente de acuerdo con apellidos y nombres pues se usarán las sentencias ORDER BY.
* Sentencia para saber la consulta de un paciente con su respectivo doctor



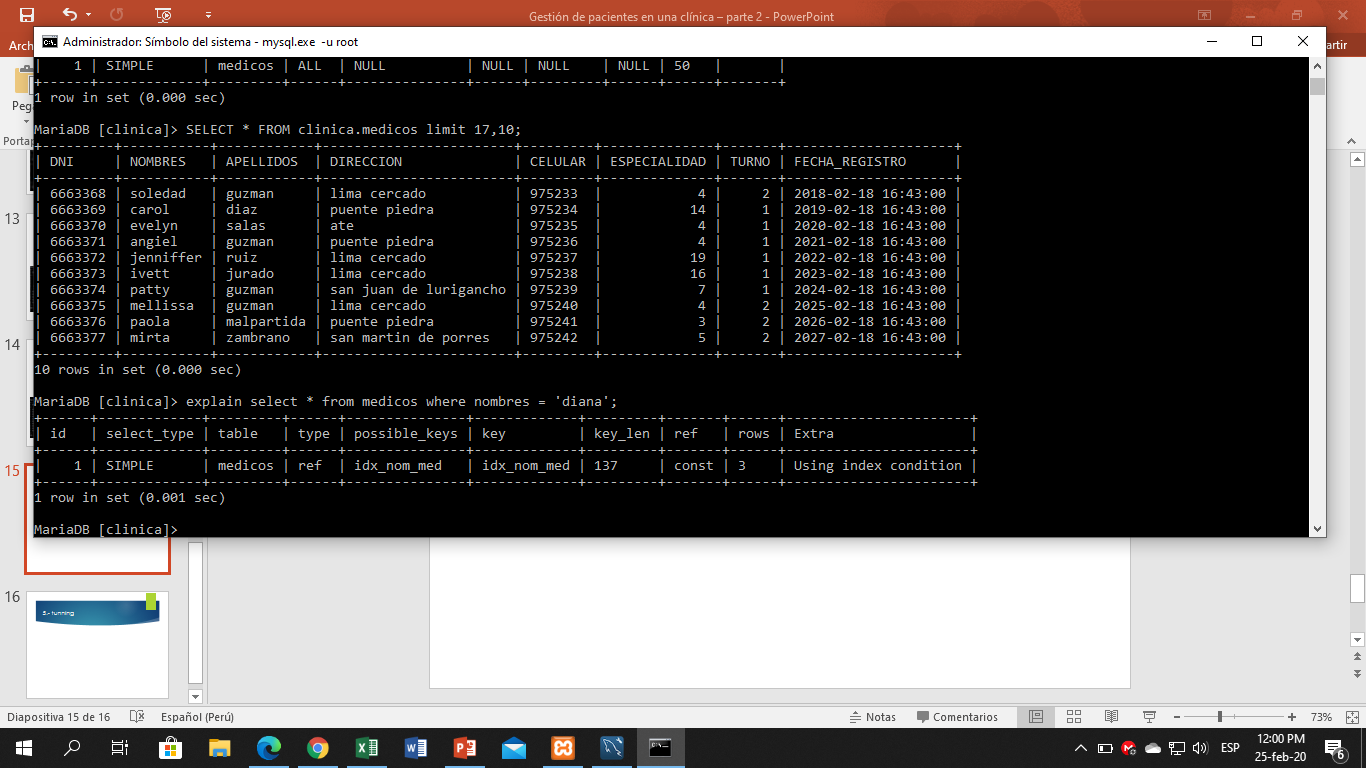
* Sentencia para saber la todos los pacientes limitando el rango de visualización a los 10 primeros.



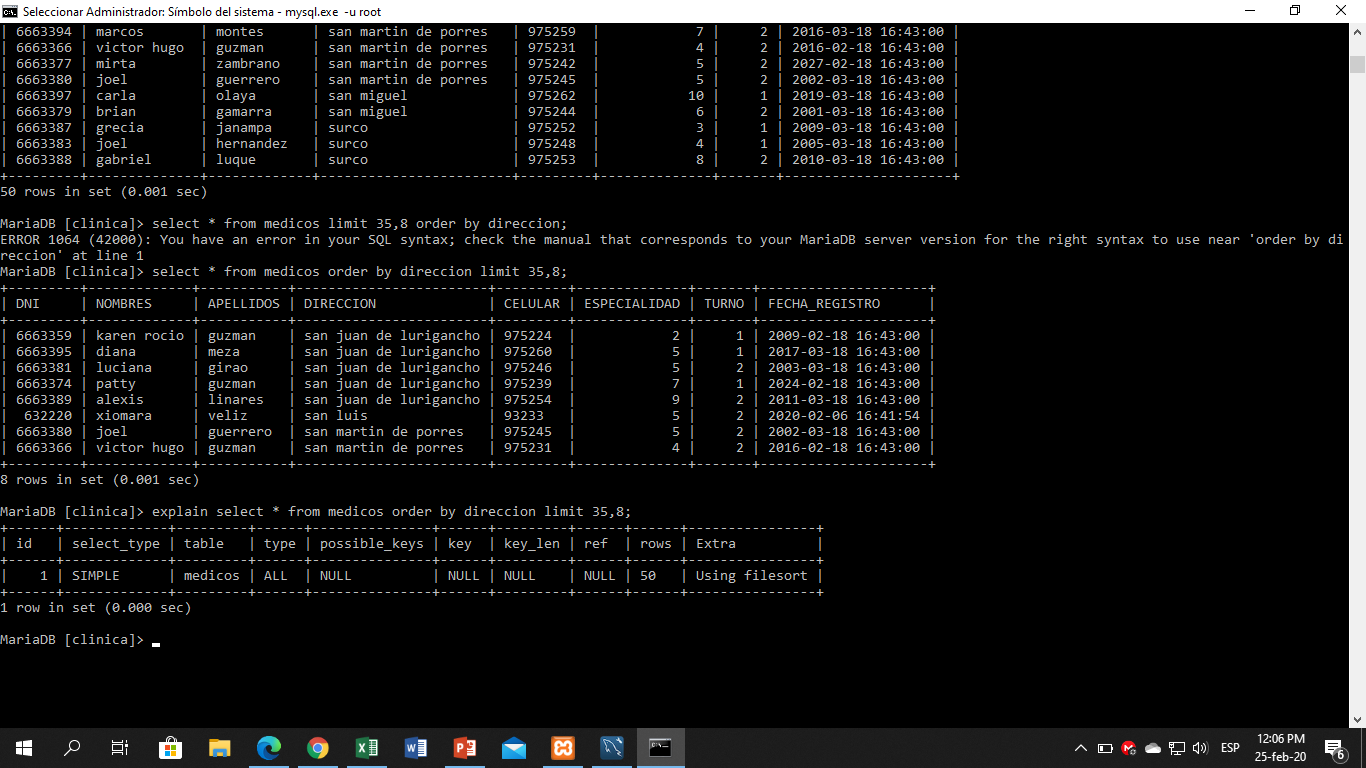
* Sentencia para saber la todos los pacientes limitando el rango de visualización a los 10 primeros a partir del 17.



* Sentencia explain al momento de buscar por nombre de doctor (hay índice en el nombre).



* Al momento de explorar los 8 primeros a partir de la fila 35 ordenados de acuerdo con su dirección. Se adjunta su sentencia explain.



# MEJORA DE RENDIMIENTO A PARTIR DEL AFINAMIENTO (TUNNING)

* Mysql Query Cache

«Query Cache» es una de las funcionalidades más interesantes que ofrece MySQL. A grandes rasgos, Query cache almacena en memoria el contenido y resultado de una consulta tipo SELECT, de modo que cuando un cliente vuelva a ejecutar la misma consulta (tiene que ser exactamente la misma), esta no tendrá que procesarse y se servirá directamente de la memoria sin necesidad de utilizar recursos de MySQL.

* + mysql> SHOW VARIABLES LIKE 'have\_query\_cache’;
  + mysql> SET GLOBAL query\_cache\_size = 40000;

# BIBLIOGRAFÍA

[1] Connolly, T & Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management*. Madrid: Editorial Pearson.

[2] Elmasri, R & Navathe, B. (2011). Fundamental of *Database Systems*. Madrid: Editorial Pearson.