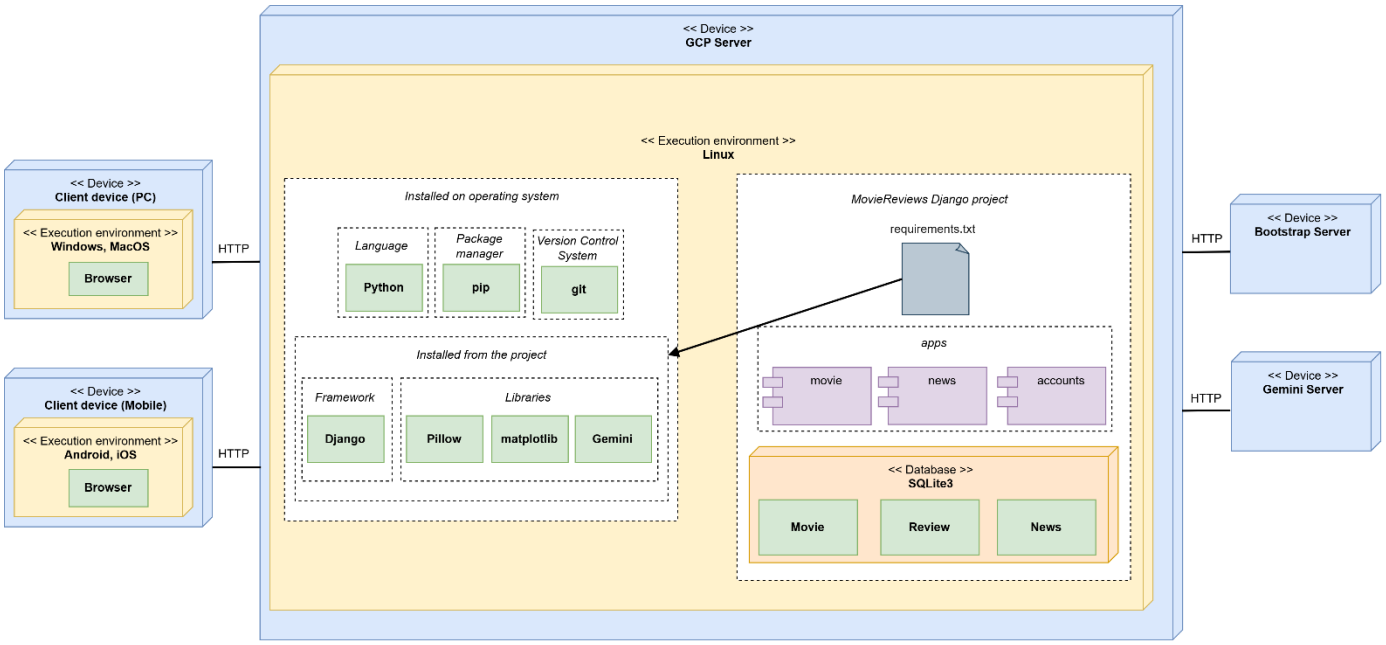
Ejemplo MovieReviews para la realización de los diagramas

Diagrama de despliegue



En el diagrama de despliegue deben mostrarse al menos dos dispositivos: el cliente (a la izquierda) y el servidor (en el centro), donde se ejecutará la aplicación.

El **dispositivo cliente** es el punto de acceso desde el cual los usuarios se conectan a la aplicación. Como se trata de una aplicación web, los usuarios pueden acceder desde computadoras y dispositivos móviles. Por ello, se debe incluir un sistema operativo adecuado para ambos tipos de dispositivos. En este caso, se ha optado por representar cada tipo de dispositivo en un nodo independiente. En cada sistema operativo debe estar instalado un navegador web, que permitirá el acceso a la aplicación.

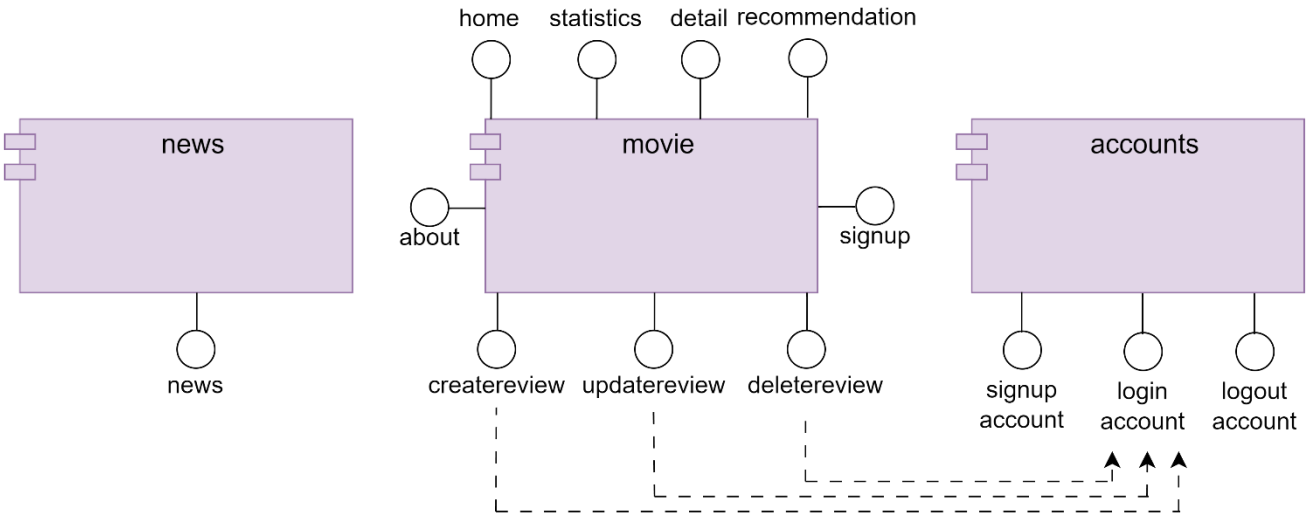
El **dispositivo servidor**, denominado *GCP Server*, es el encargado de alojar la aplicación. Dado que se desplegará en Google Cloud Platform (GCP), se debe especificar que el sistema operativo es Linux. Además, es necesario listar los elementos que deben instalarse en el servidor: el lenguaje de programación, el gestor de paquetes, el sistema de control de versiones y los componentes del proyecto, como el framework de desarrollo web y las bibliotecas necesarias.

También se deben incluir las **apps del proyecto**, que corresponden a los módulos de la aplicación: *movie*, *news* y *accounts*. Asimismo, se debe indicar el archivo que contiene la lista de dependencias del proyecto.

Finalmente, es necesario especificar la base de datos utilizada, en este caso SQLite3, junto con las tablas que se crearán en la aplicación web: *Movie*, *Review* y *News*.

Los dispositivos *Gemini Server* y *Bootstrap Server* se representarán sin detalles internos debido a la falta de información sobre su configuración.

Diagrama de componentes



El diagrama de componentes debe especificar claramente las funciones que ofrece cada componente, ya que cada función representa un requisito funcional del proyecto. Este diagrama ilustra la distribución de los requisitos funcionales entre los distintos componentes y sus interrelaciones.

Hay tres componentes: *movie*, *news* y *accounts*, junto con sus conexiones. El componente *movie* contiene las funciones:

- Presentar la página principal (*home*).
- Mostrar la página *about* (*about*).
- Visualizar gráficas estadísticas (*statistics*).
- Mostrar detalles de las películas (*detail*).
- Gestionar *reviews*, permitiendo su creación (*createreview*), actualización (*updatereview*) y eliminación (*deletereview*).
- Mostrar la página de registro (*signup*).

El componente *news* tiene una función: mostrar las noticias (*news*).

El componente *accounts* gestiona funciones relacionadas con la autenticación de usuarios:

- Registro (*signup*).
- Inicio de sesión (*login*).
- Cierre de sesión (*logout*).

Además, el diagrama muestra que el componente *movie* depende de la funcionalidad *login* del componente *accounts* para permitir la creación, actualización y eliminación de *reviews*. Esta dependencia se representa con líneas punteadas dirigidas a la función correspondiente.

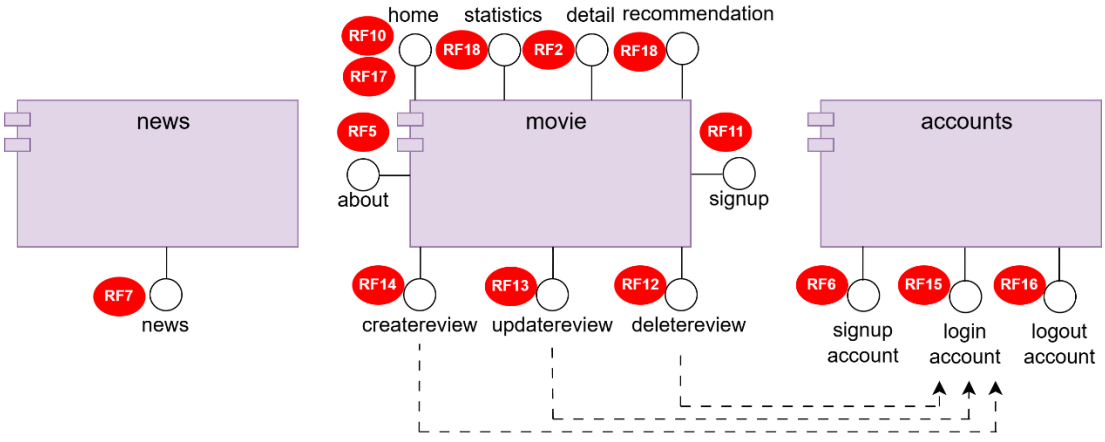
Por último, todas las funcionalidades representadas en este diagrama deben estar explícitamente disponibles en la página web a través de botones, menús u otras opciones de navegación.

Requisitos funcionales

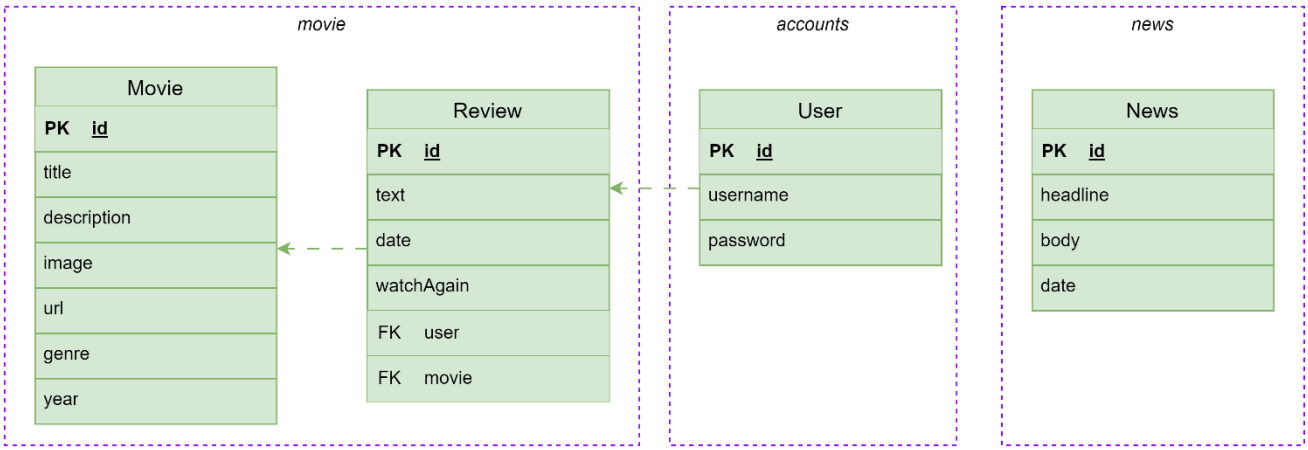
La siguiente tabla presenta la relación entre los requisitos y las funciones de los componentes.

Id	Requisito	Componente	Función
FR7	The system should display the most recent news on the news page	news	news
FR5	The system could display information about its developers	movie	about
FR10	The Movie Reviews application shall display all movies	movie	home
FR17	The Movie Reviews application should provide the user with the ability to search movies by name	movie	home
FR18	The Movie Reviews application should provide the user with the ability to search movies by semantic similarity	movie	recommendation
FR18	The system should display graphics about genre and year of movies	movie	statistics
FR2	While a user is on the movie details page, the system shall display the most recent reviews for that movie	movie	detail
FR11	The Movie Reviews application shall display a basic signup page	movie	signup
FR12	If **the user is logged in**, then the MovieReviews application provide the user with the ability to delete a movie review	movie	deletereview
FR13	If **the user is logged in**, then the MovieReviews application provide the user with the ability to update a movie review	movie	updatereview
FR14	If **the user is logged in**, then the MovieReviews application provide the user with the ability to create a movie review	movie	createreview
FR6	The web application shall support user authentication including sign-up functionality	accounts	signupaccount
FR15	The Movie Reviews application shall provide the users with the ability to login to the application	accounts	loginaccount
FR16	The Movie reviews application shall provide the users with the ability to logout from the application	accounts	logoutaccount

El siguiente diagrama ilustra la relación entre los requisitos y las funciones del diagrama de componentes.



Modelo de Datos



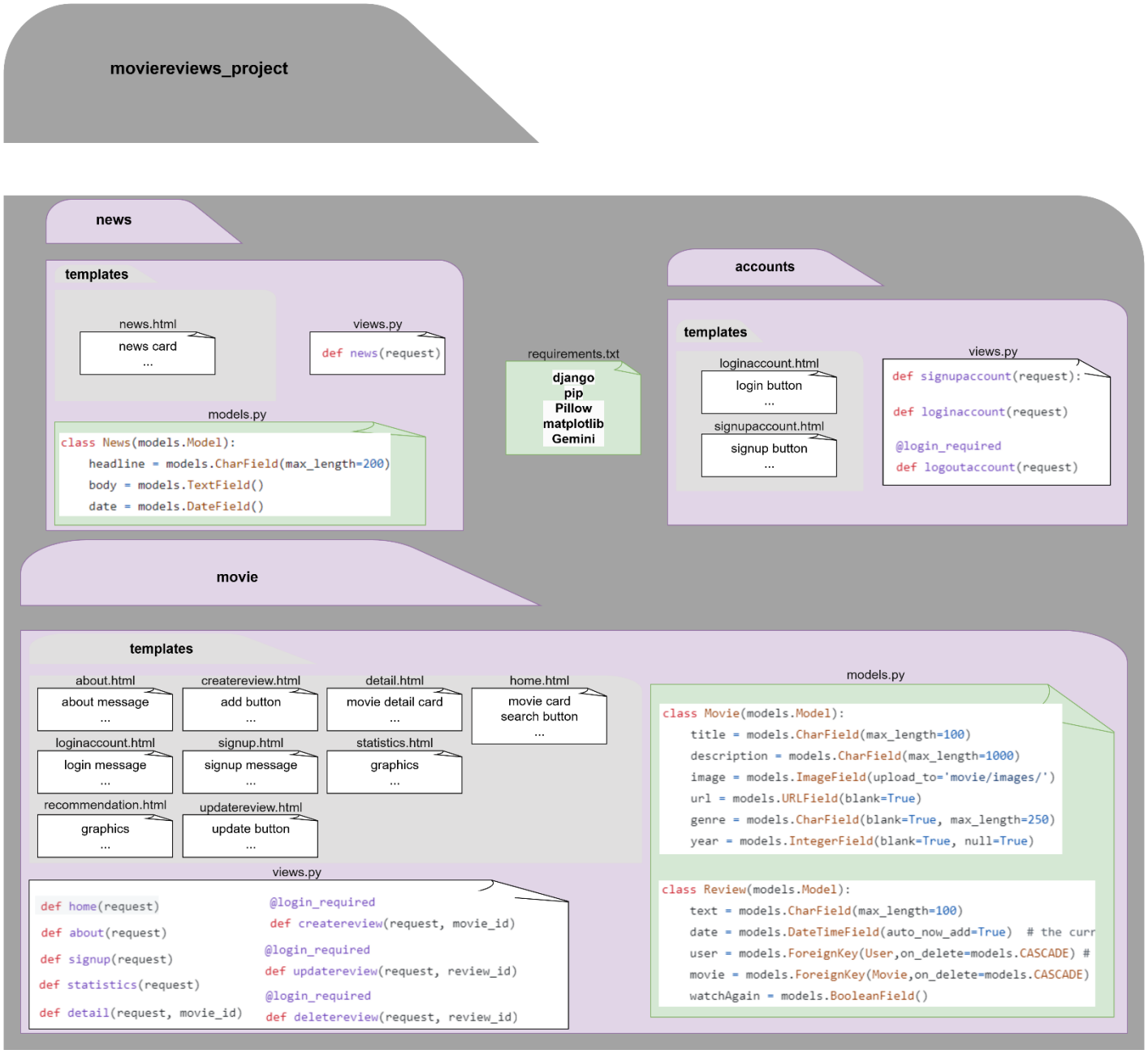
El diagrama muestra tres entidades: **Movie**, **Review** y **News**.

- **Movie** representa las películas y tiene los atributos: *id* (clave primaria), *title* (título), *description* (descripción), *image* (imagen), *url* (enlace), *genre* (género) y *year* (año). Cada película puede tener múltiples reseñas.
- **Review** representa las reseñas de películas. Sus atributos incluyen: *id* (clave primaria), *text* (texto de la reseña), *date* (fecha de creación), *watchAgain* (indicación de si el usuario volvería a ver la película), *movie* (clave foránea que referencia a *Movie*) y *user* (clave foránea que representa al usuario que escribió la reseña). Aunque en este ejemplo se usa el modelo de usuarios de Django, la entidad *User* se representa explícitamente. Cada reseña está asociada a una única película.
- **News** representa noticias y tiene los atributos: *id* (clave primaria), *headline* (título de la noticia), *body* (cuerpo de la noticia) y *date* (fecha de publicación). Esta entidad no tiene relaciones directas con otras en el diagrama.

La relación principal en este modelo es entre *Movie* y *Review*: una película puede tener múltiples reseñas, pero cada reseña pertenece a una sola película. Además, la flecha entre *Review* y *User* indica que cada reseña está vinculada a un usuario específico, permitiendo que un usuario tenga varias reseñas.

Los recuadros punteados muestran a qué componente pertenece cada entidad dentro del sistema.

Implementación



Este diagrama representa la estructura del proyecto Django y su correspondencia con el diagrama de componentes. Se observa que el proyecto consta de tres aplicaciones, una por cada componente: *news*, *movie* y *accounts*.

En el archivo *views.py* de cada aplicación debe haber al menos una función que implemente cada requisito funcional representado en el diagrama de componentes. En los archivos *HTML* de cada app deben estar, como mínimo, las interfaces de las funcionalidades a las que el usuario puede acceder directamente.

Por otro lado, en el archivo *models.py* de cada app se definen los modelos que se convertirán en tablas de la base de datos. De este modo, se establece una conexión directa entre los requisitos, los componentes y la estructura del proyecto Django.