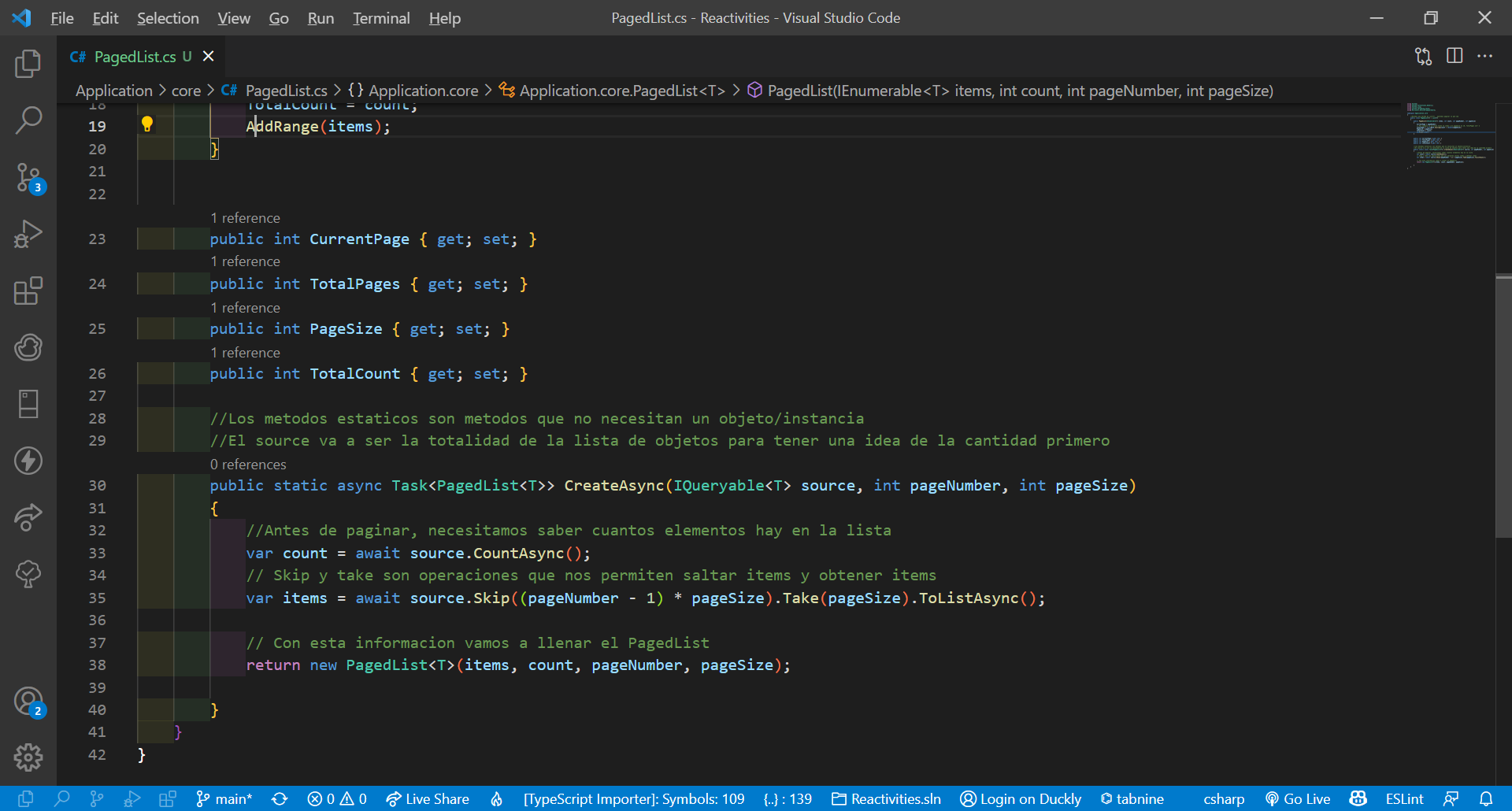
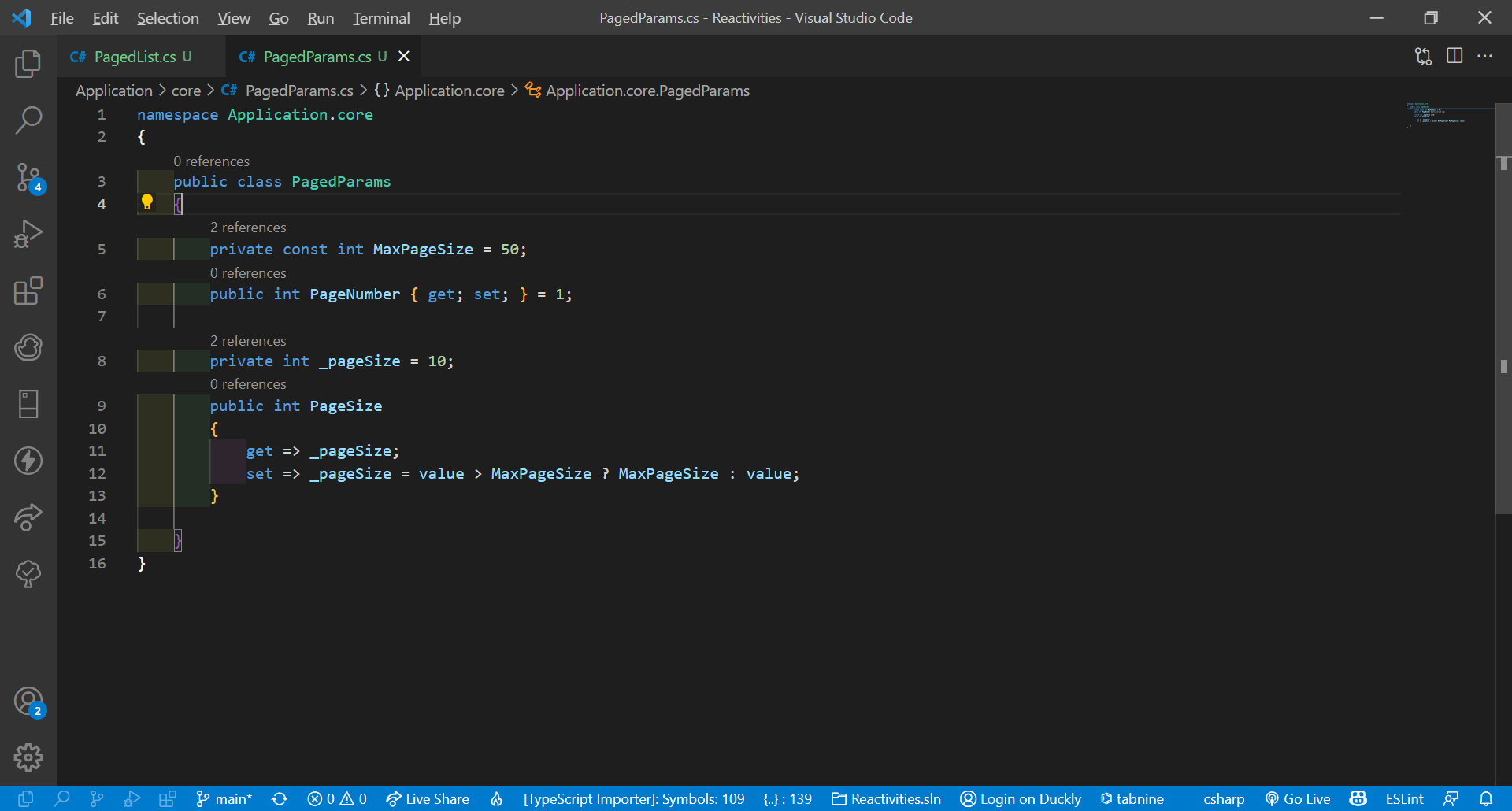
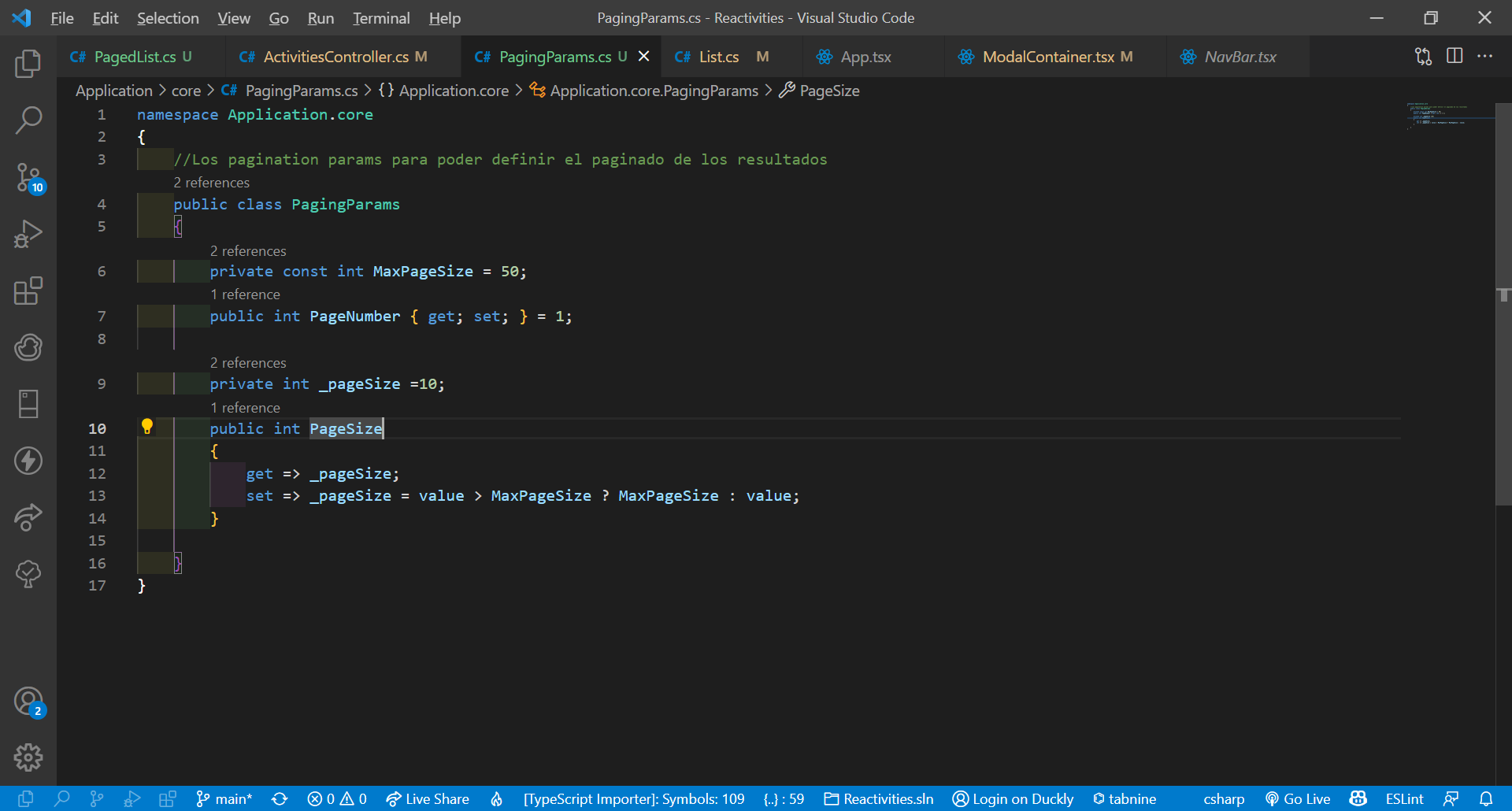
Vamos a hacer un pagination al estilo de Twitter. O sea que se vaya cargando a medida que scrolleamos.

Creamos para eso la carpeta PagedList en Application->Core

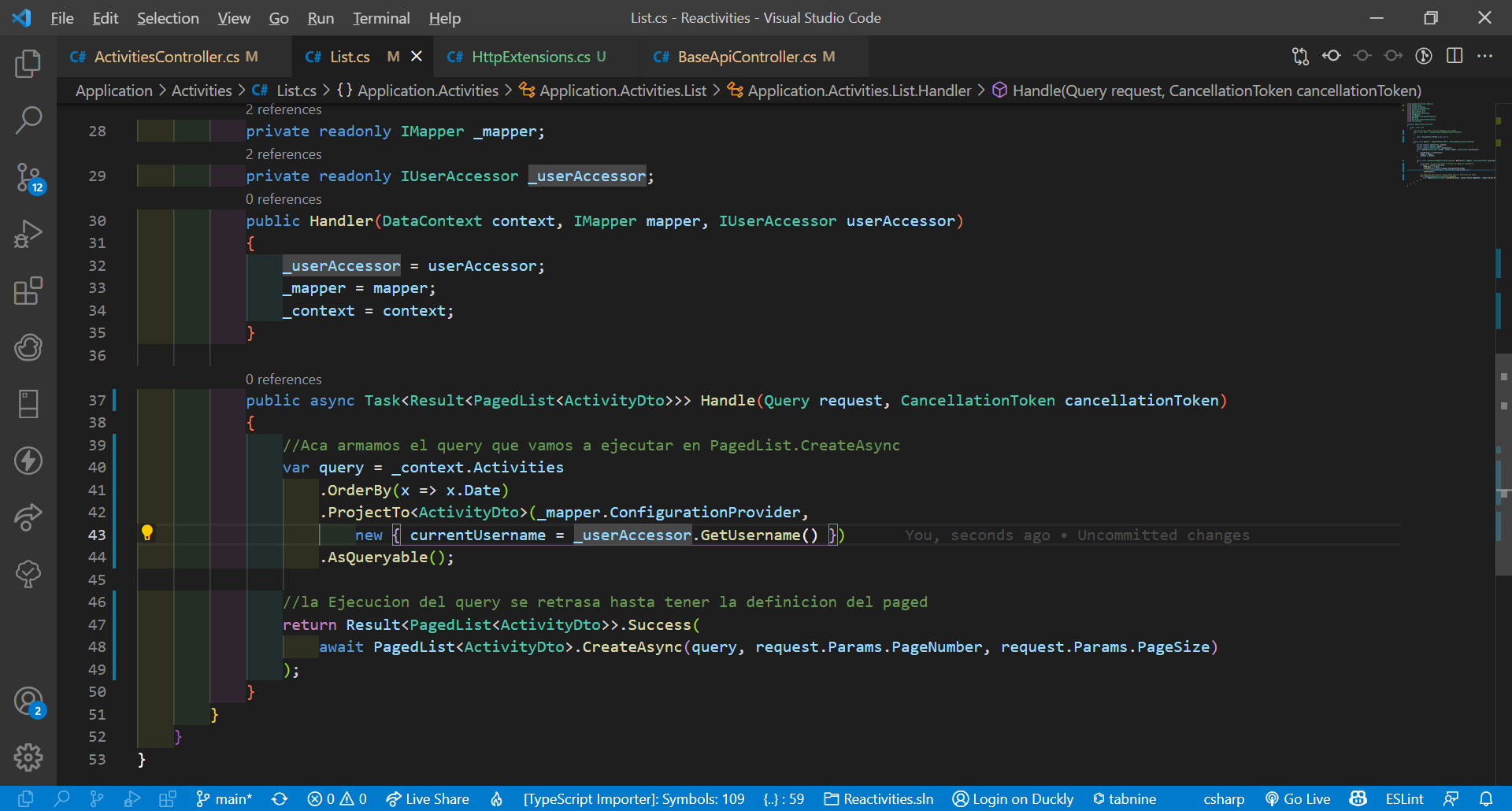


Creamos el Paged Params

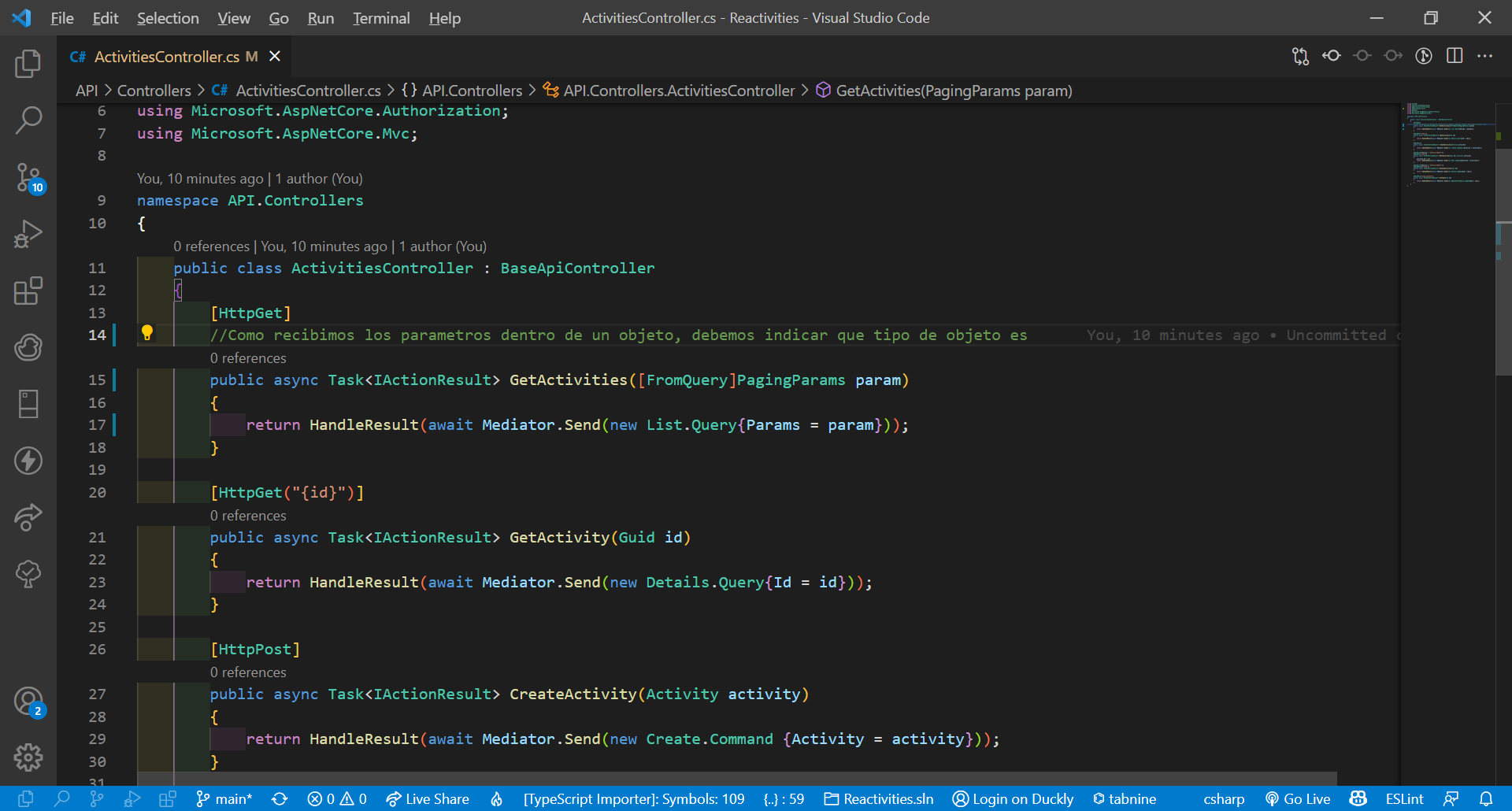




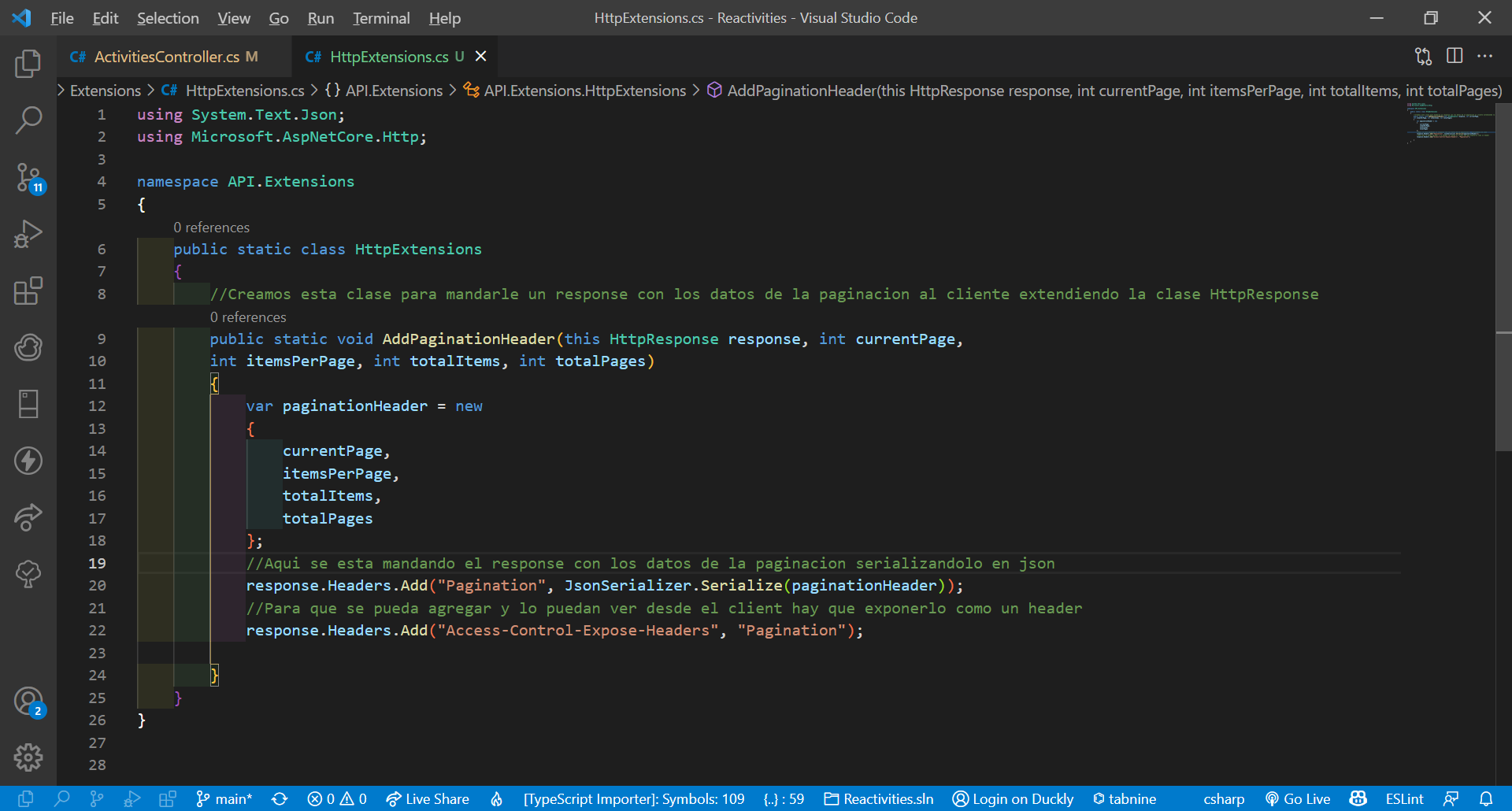
Modificamos la clase List para usar la PagedList y pasarle los paginparams



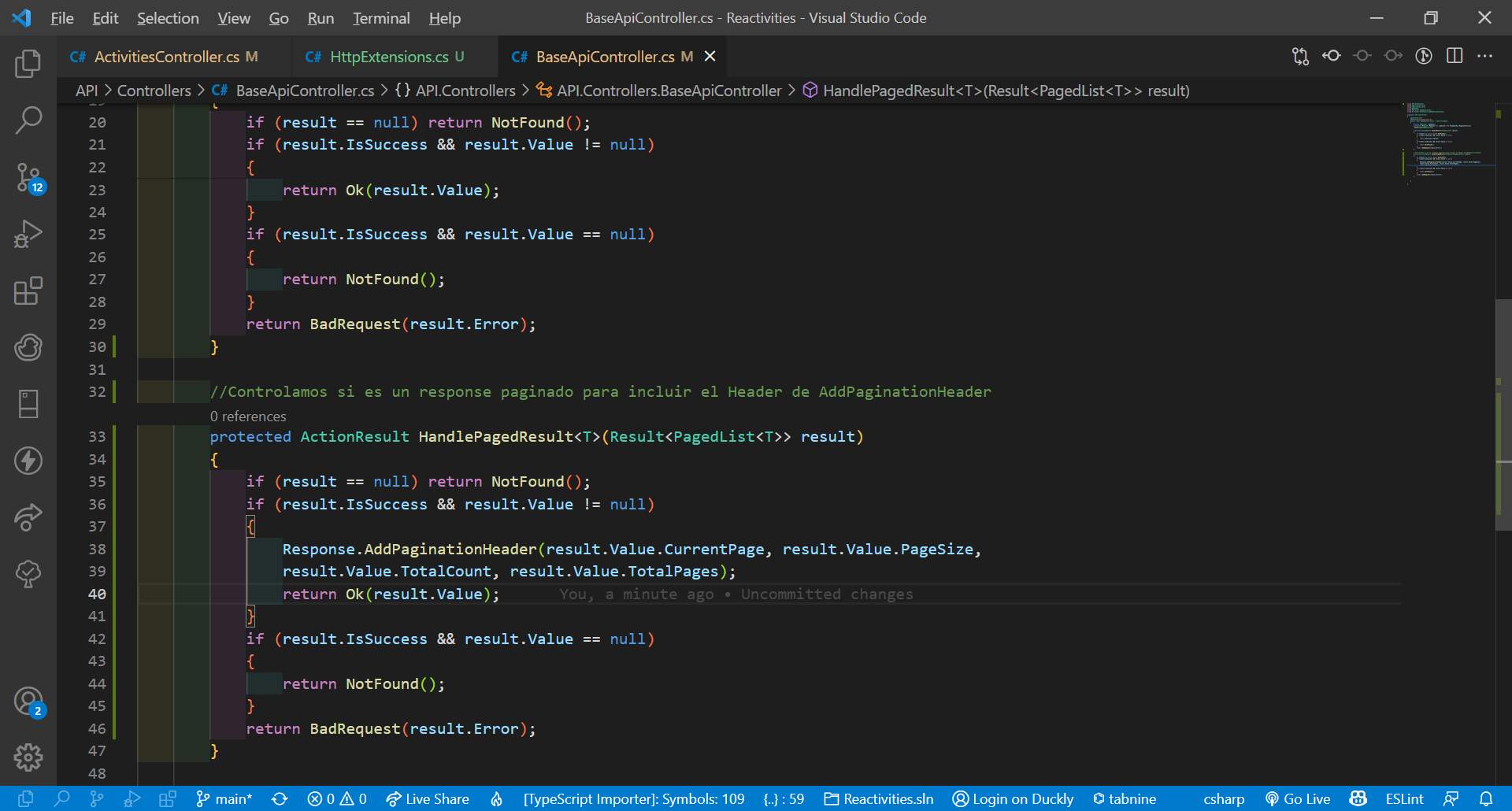
Modificamos el ActivitiesController

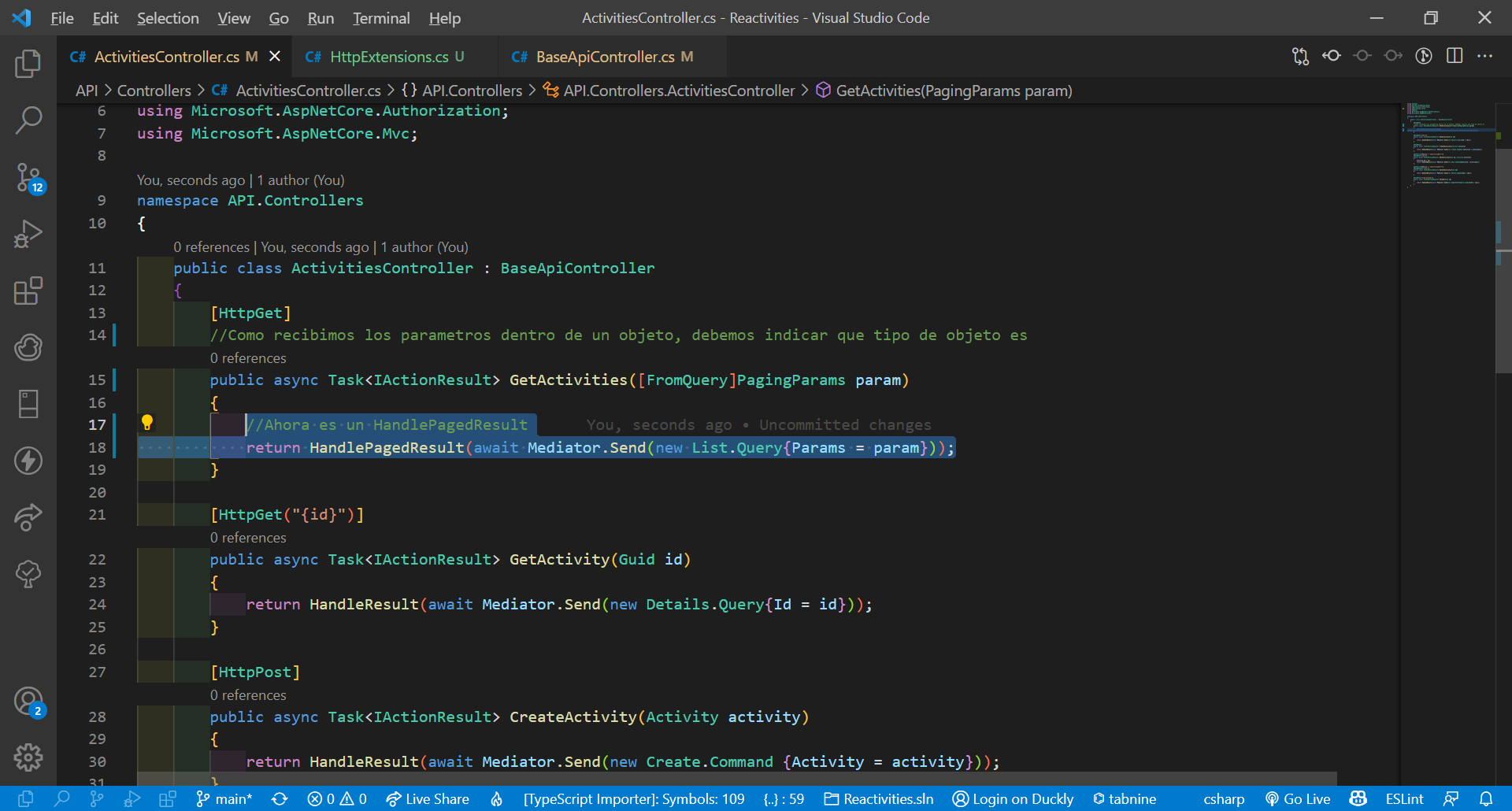


Para enviarle al cliente los datos de la paginación actual tenemos que extender la clase httpResponse, lo hacemos creando la clase HttpExtensions y luego exponiéndola. Luego debemos meter esta clase en cada resultado que queremos paginar

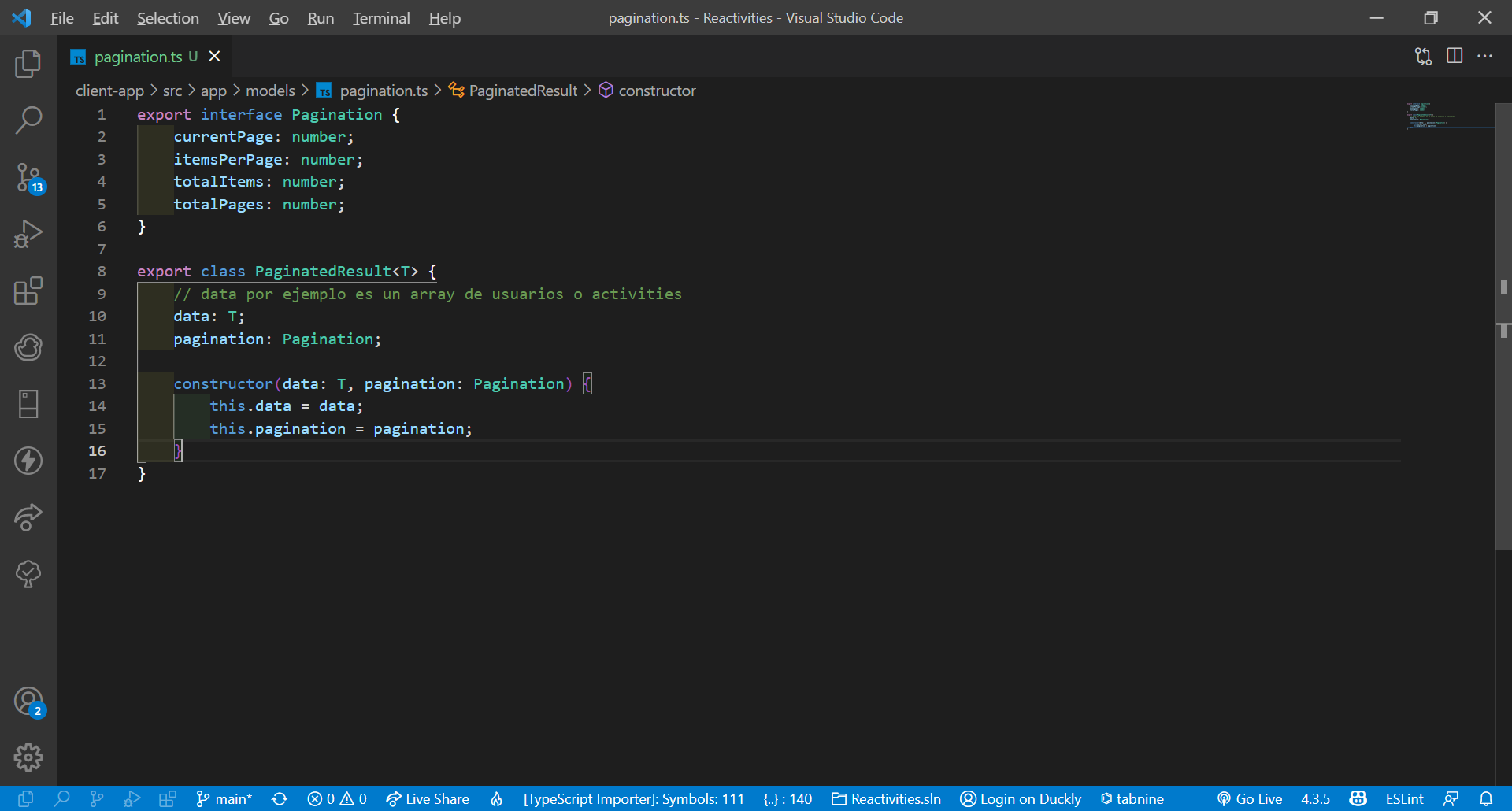


En BaseApi creamos un tratamiento para cuando se devuelven PagedList para poder agregar el Header

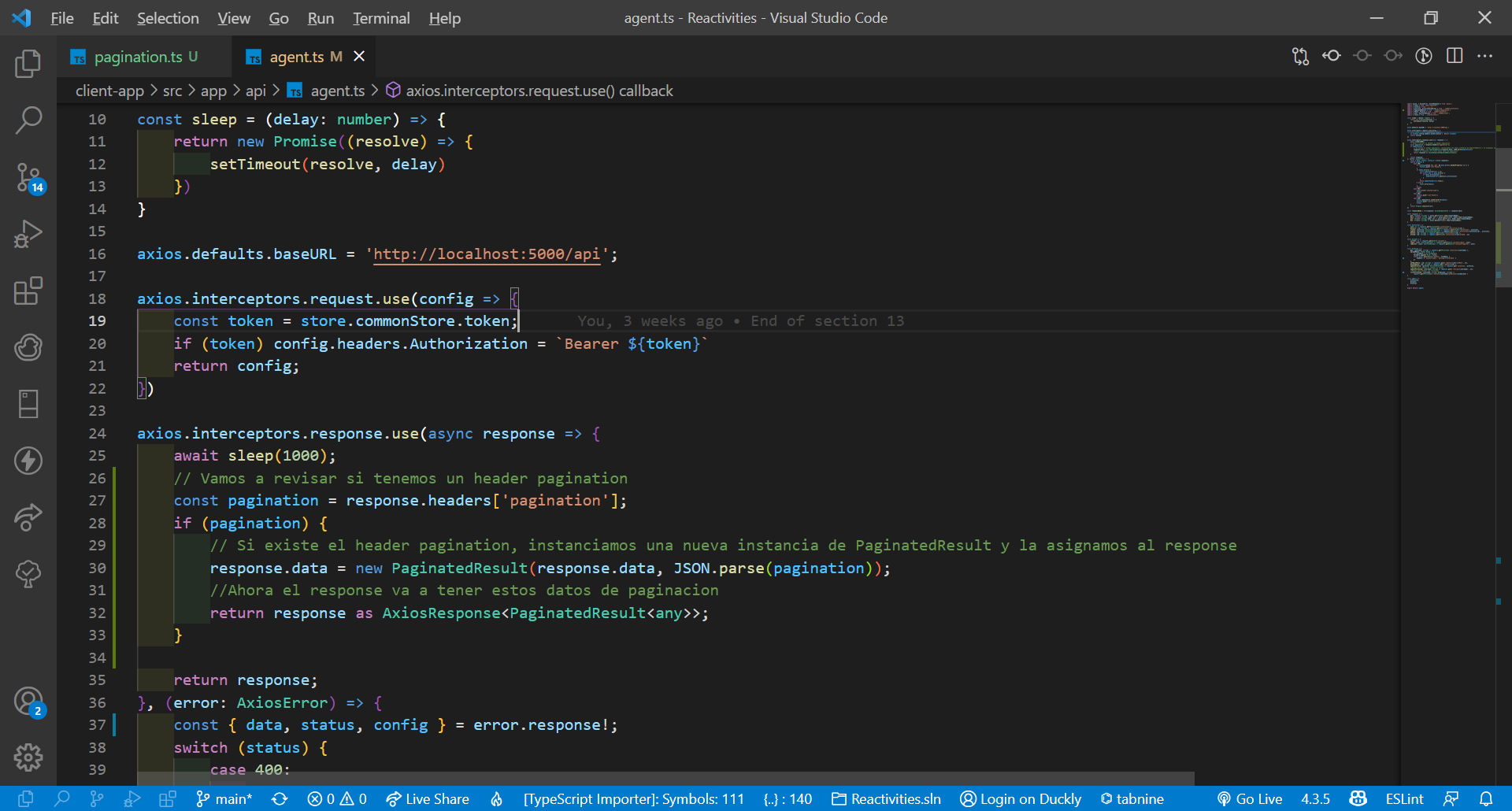




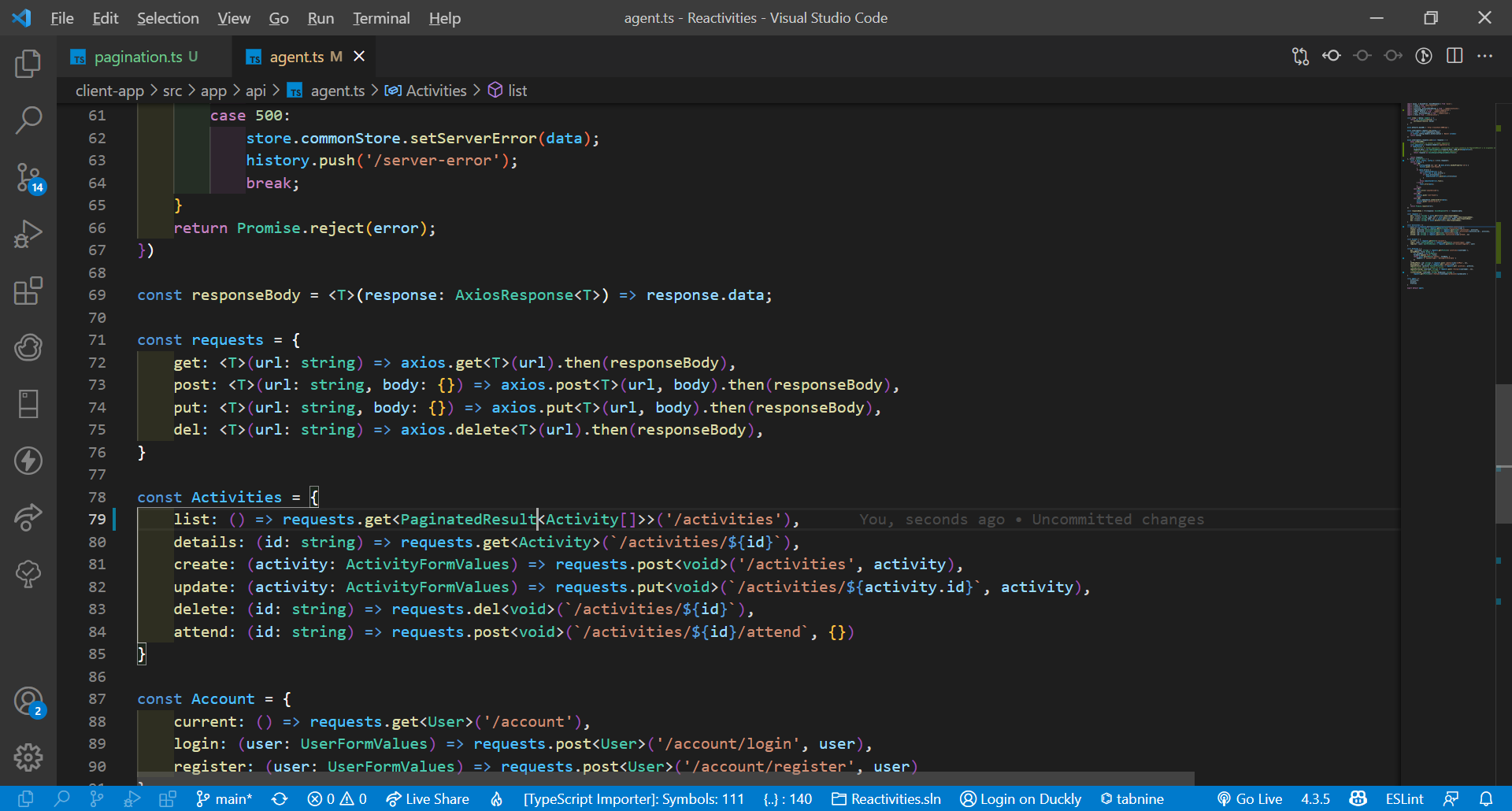
Para recibir y enviar la paginación desde el cliente creamos una interface pagination y un clase PaginatedResult



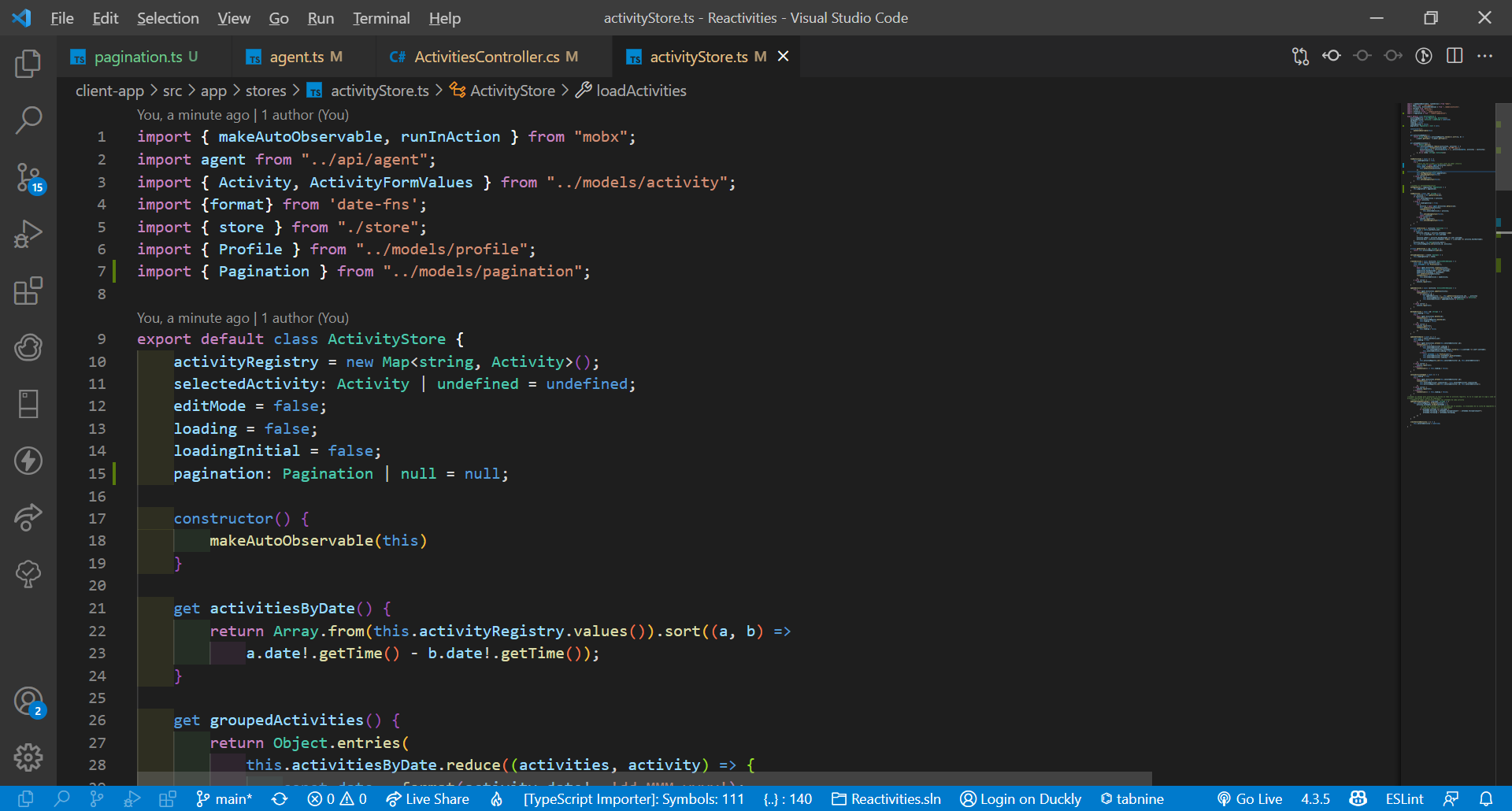
Modificamos el interceptor del response en agent.ts para corroborar si tiene un header “Pagination” y devolver un PaginatedResult

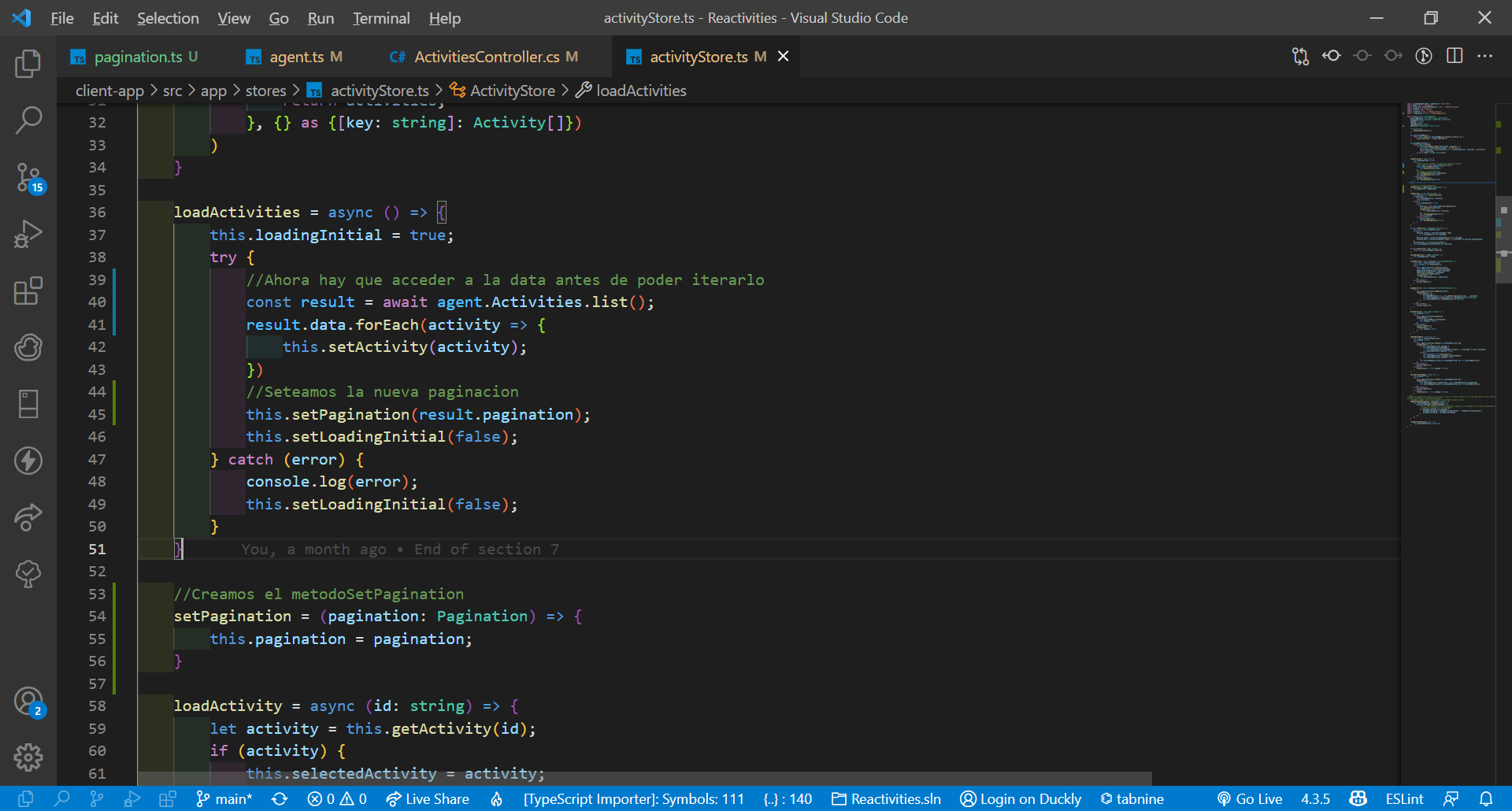


Aclaramos que vamos a recibir un paginatedResult



Modificamos el activitiStore para paginar y setear la paginación

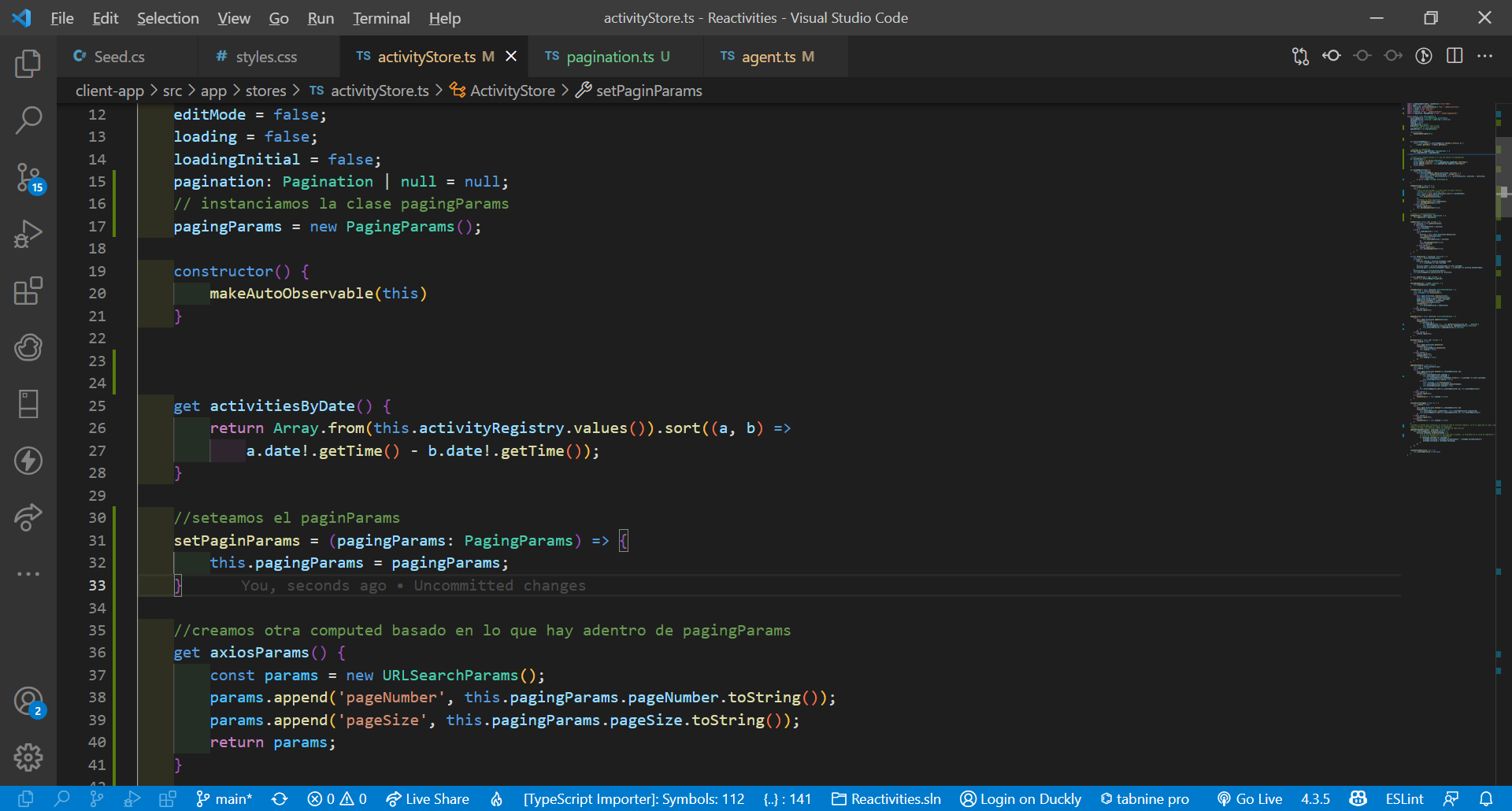


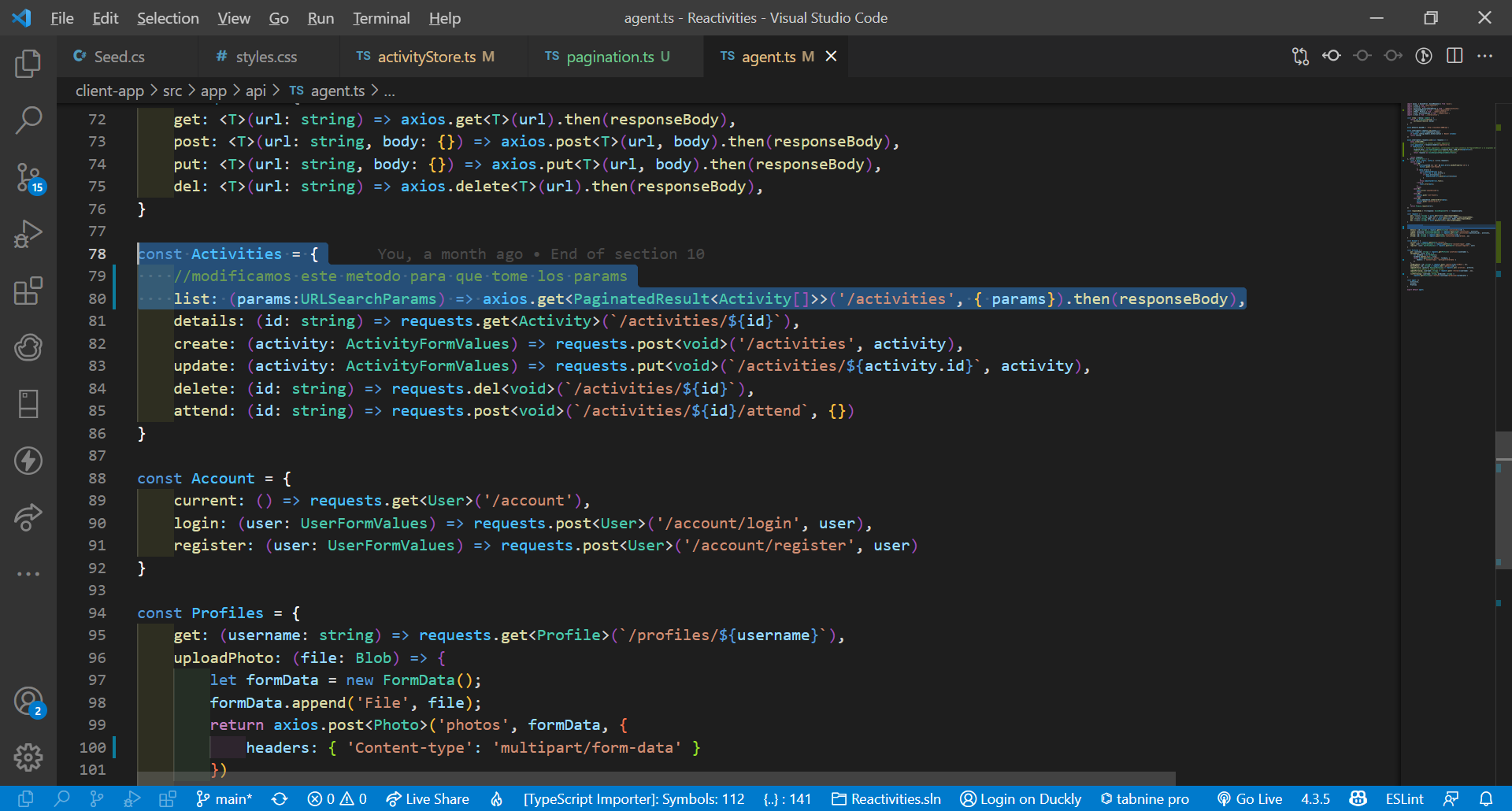


-Creamos un paginParams para pedir las paginas y elementos al servidor

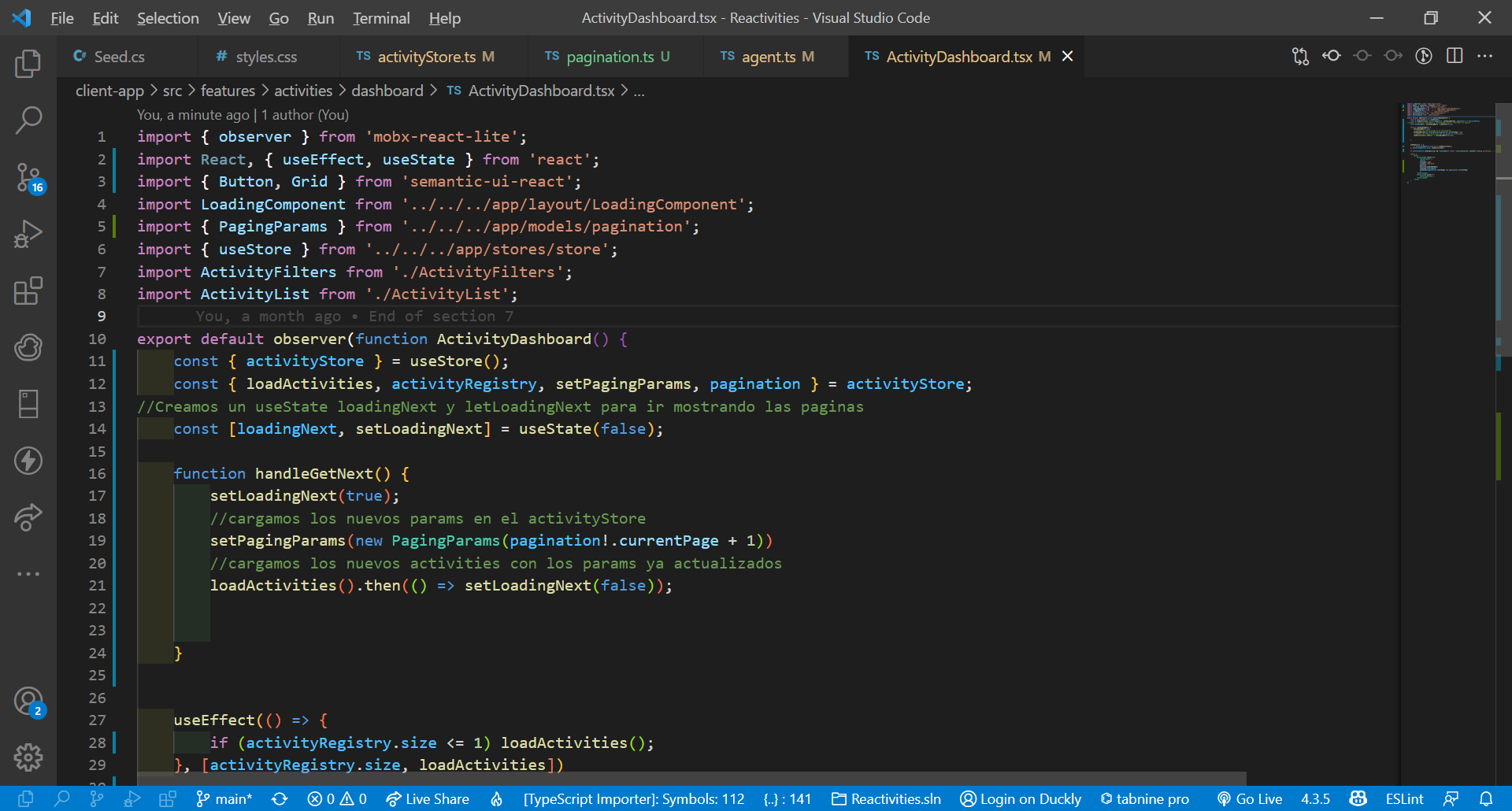


-En el activityStore instanciamos esa clase, luego creamos un setAxiosParams para poder llenarla y un computed para poder pasarlo como parámetro al método list del agent.ts el que antes tenemos que modificar



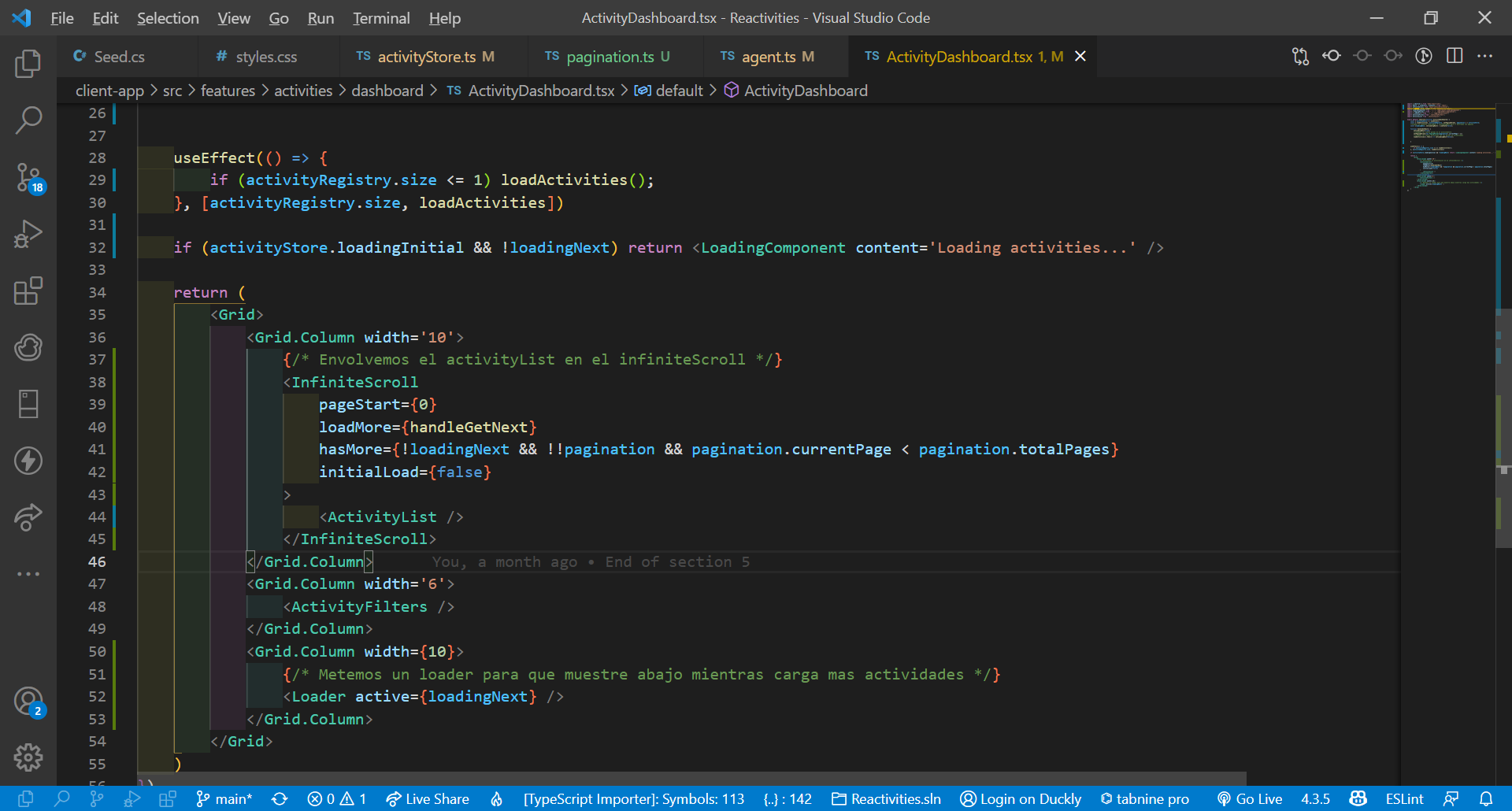


Actualizamos el activityDashboard para ir mostrando mas actividades con los métodos que fuimos creando

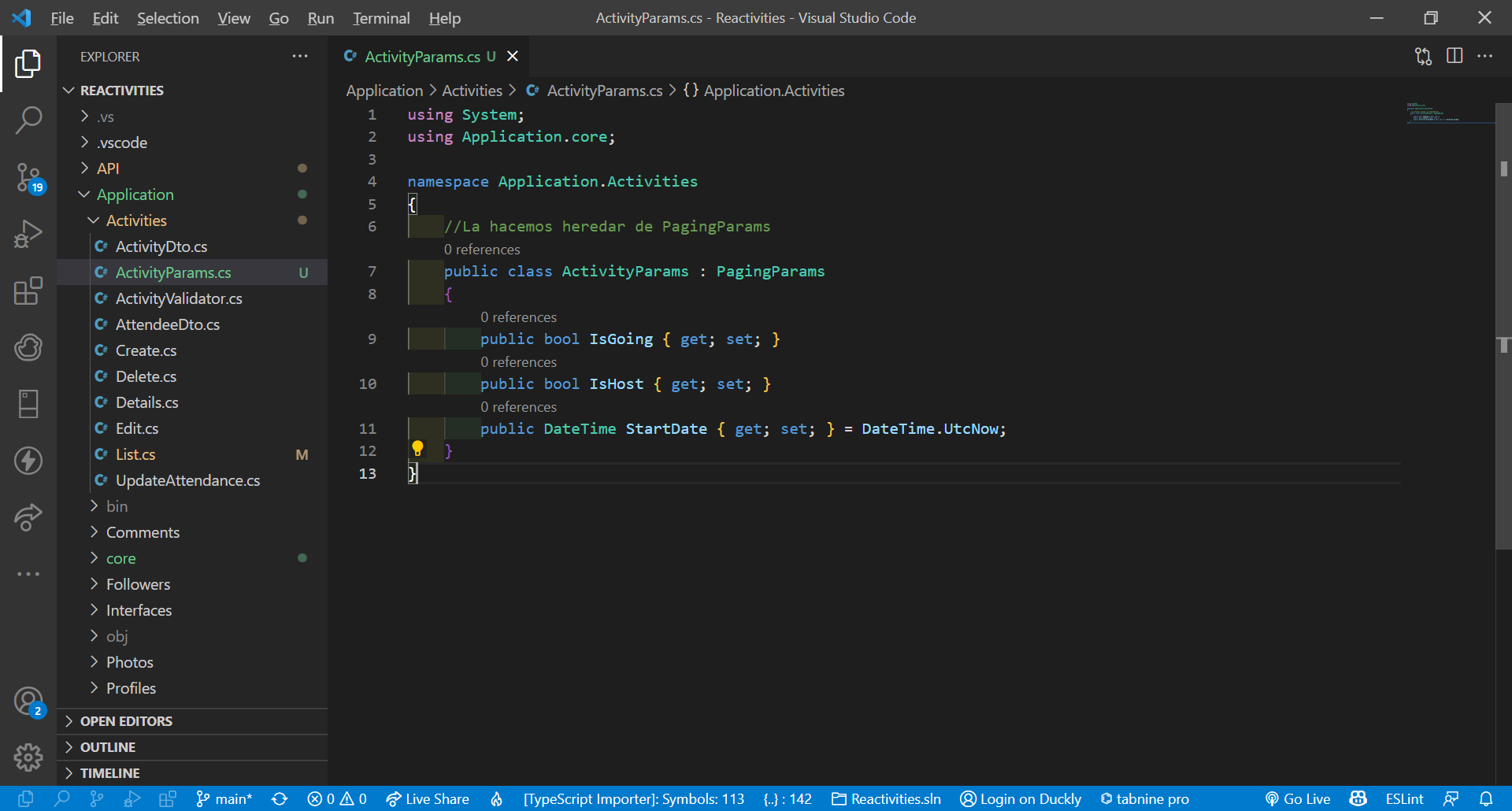


Para el infinite scroller necesitamos instalar el react-infinite-scroller

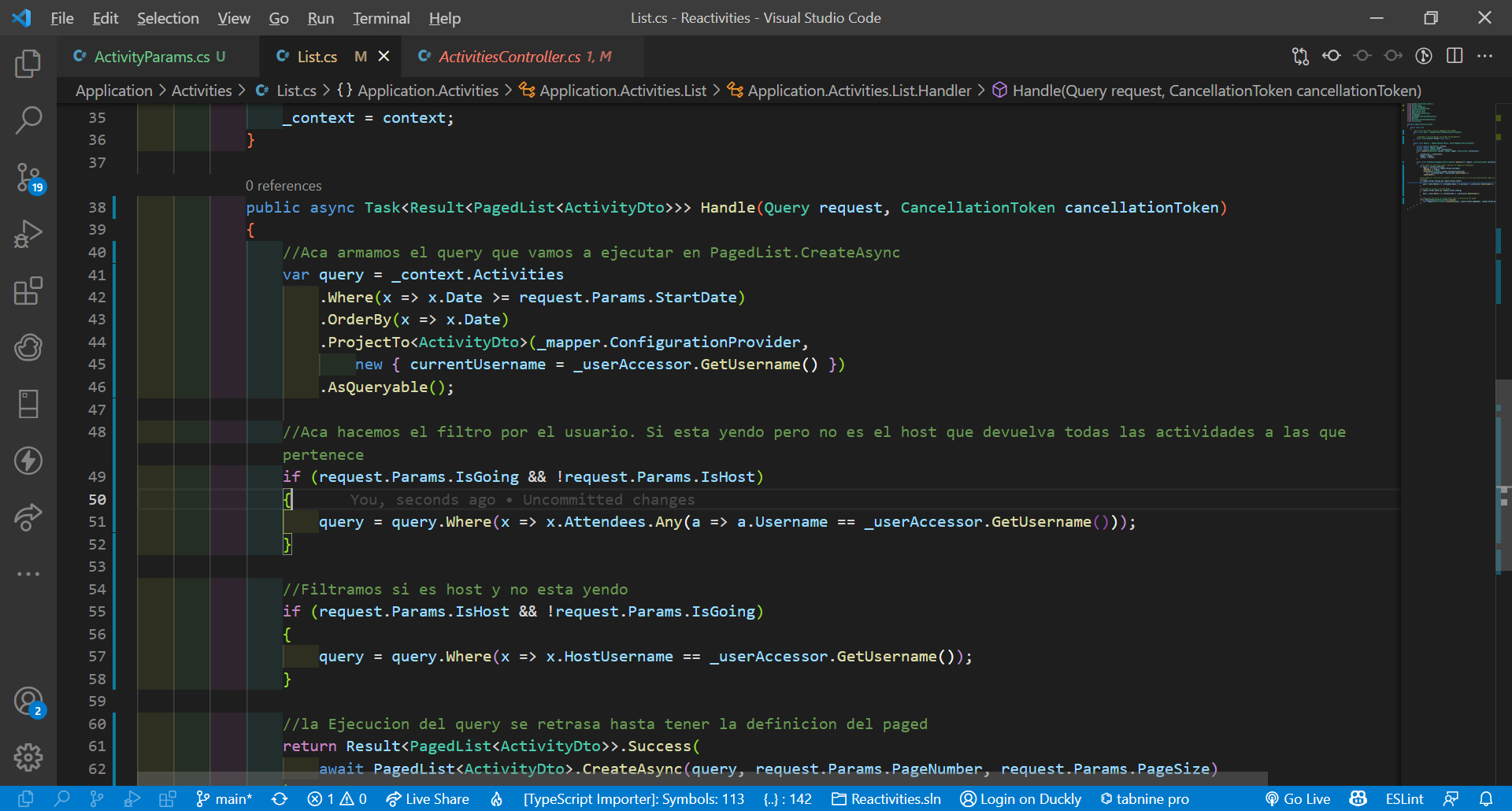
También necesitamos instalar el type definition file



Para hacer un filter mas completo con params creamos una clase AtivityParams y hacemos que herede de PAgedParams



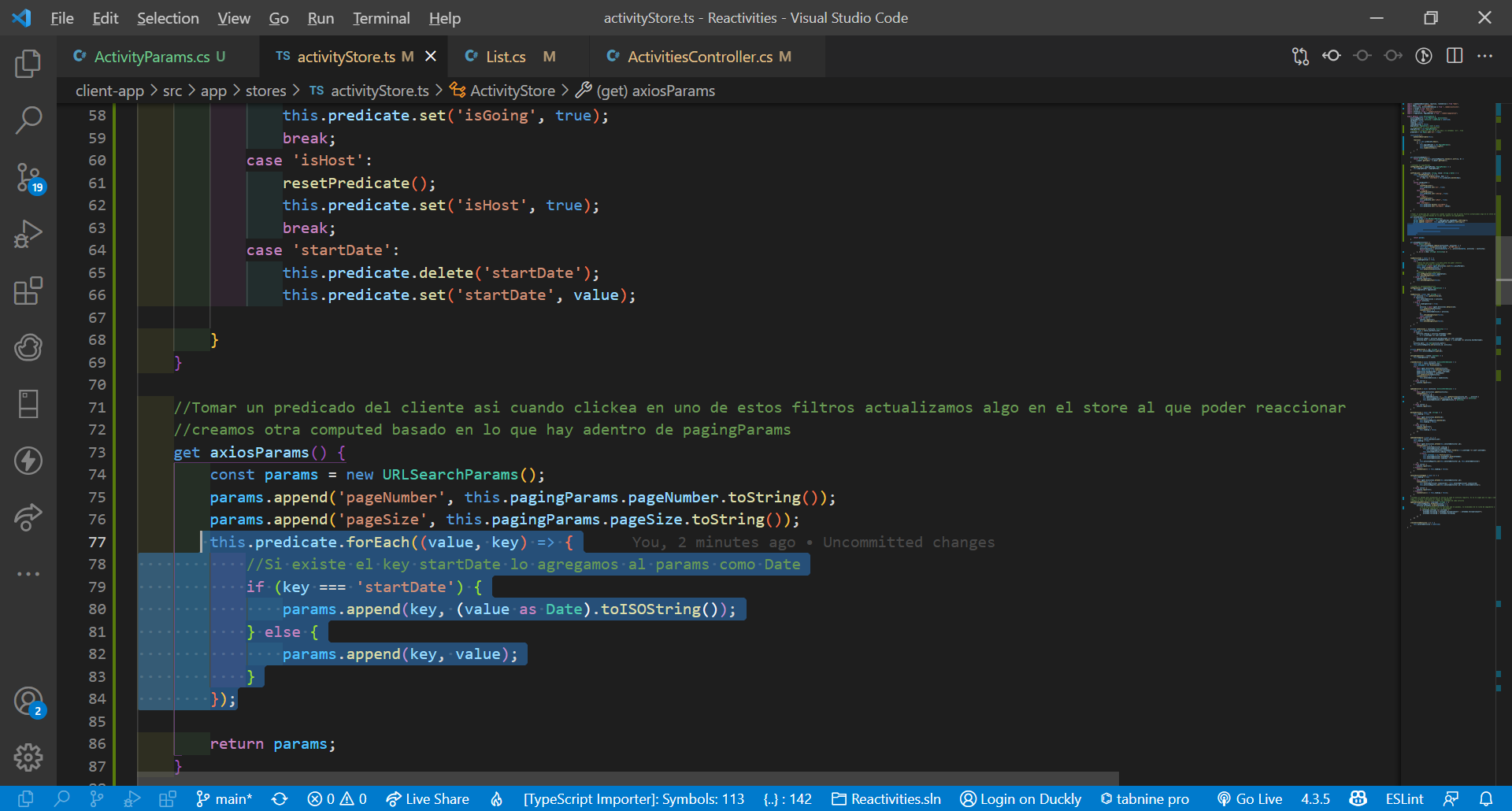
Modificamos el List para los filtros. Tambien cambiamos lo que recibe.Ahora recibe ActivityParams



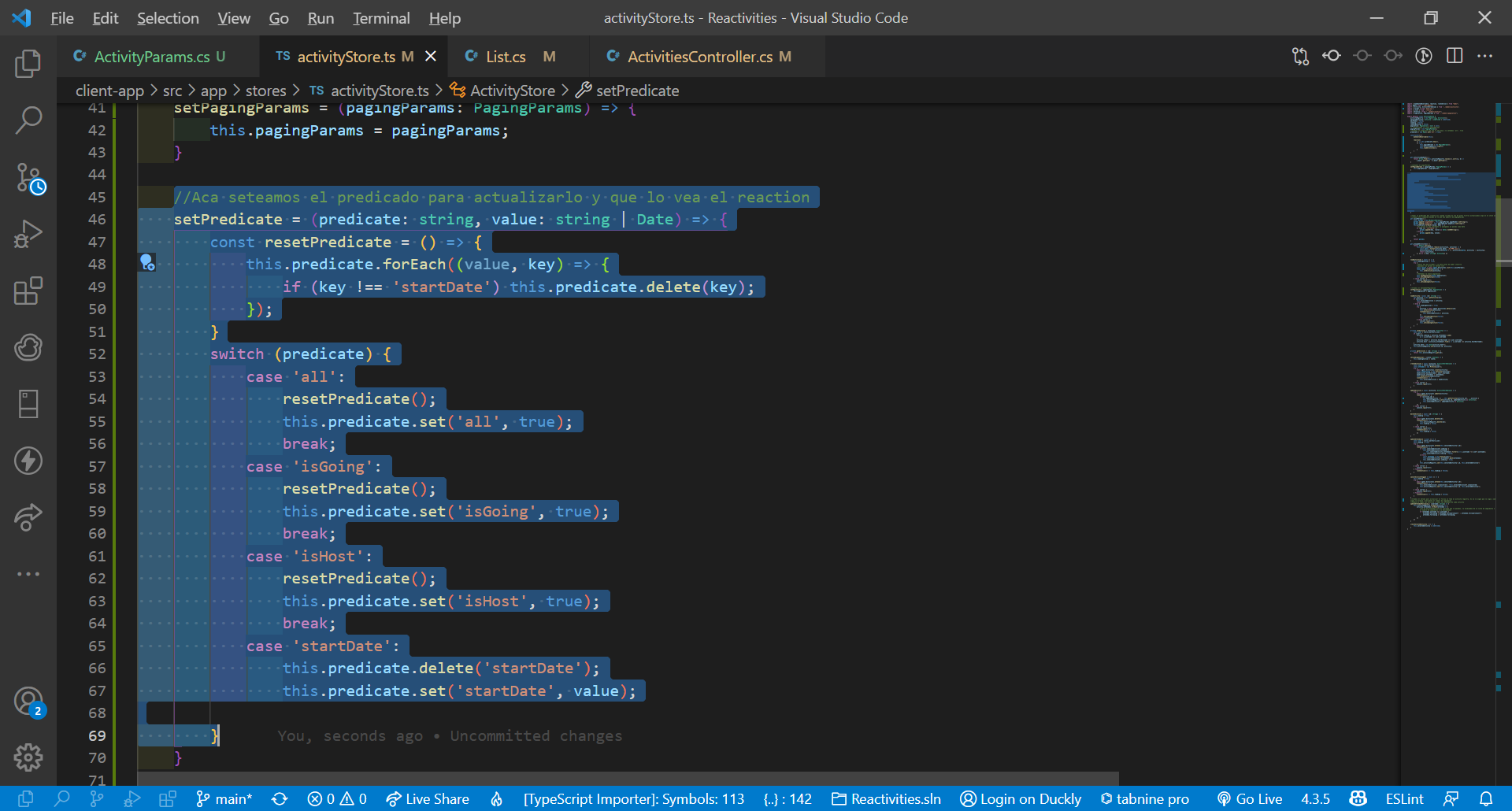
Actualizamos el Controller



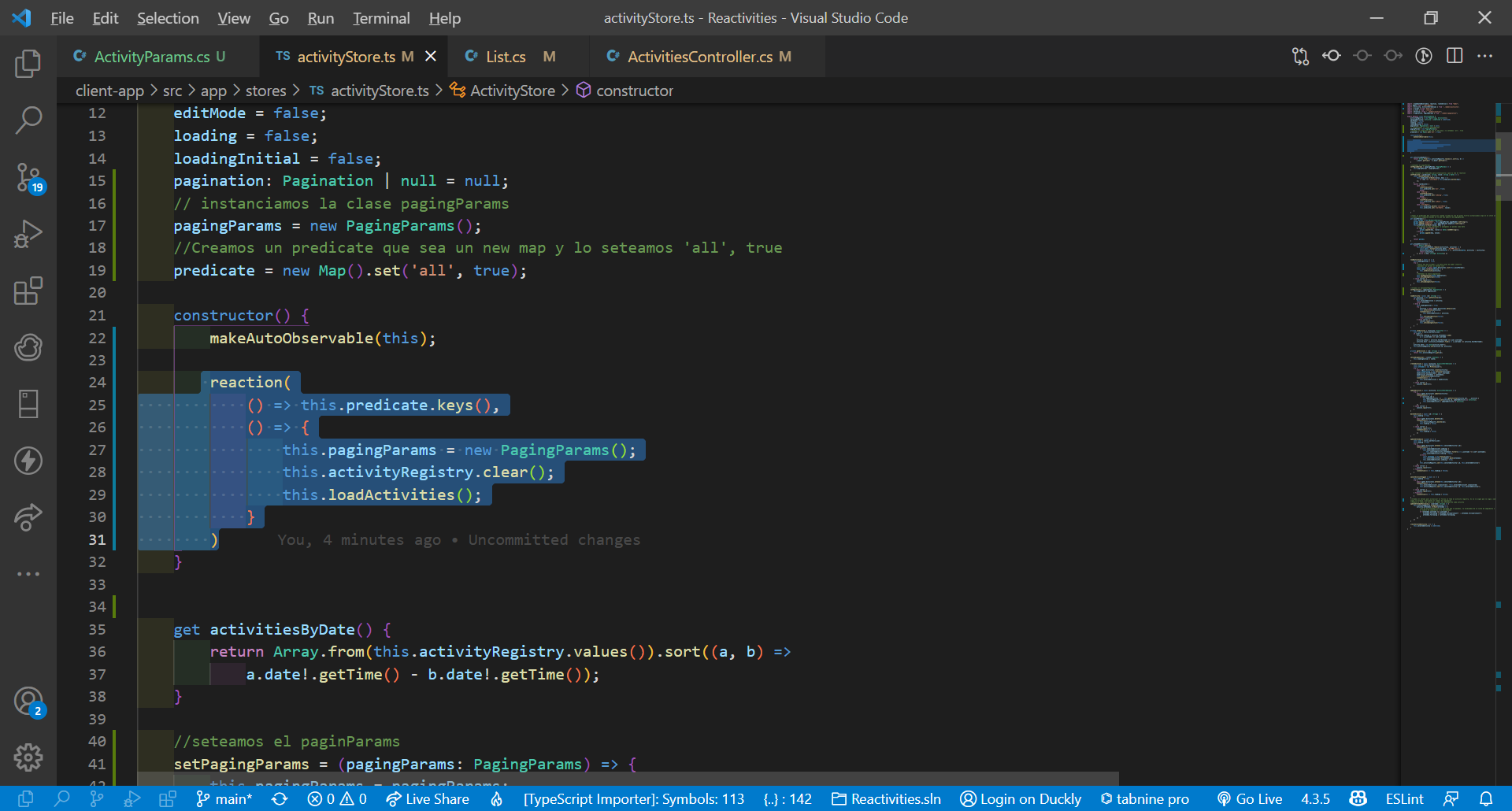
Tomamos un predicado del cliente asi cuando clickea en uno de estos filtros actualizamos algo en el store al que poder reaccionar. Si el key es stardate lo pasamos Date



Tenemos que hacer una función para cambiar el predicado y que lo vea el Mobx.reaction



Creamos el reaction para que este mirando los cambios de los keys



Cambiamos el activityFilters para que tome y setee los predicados

