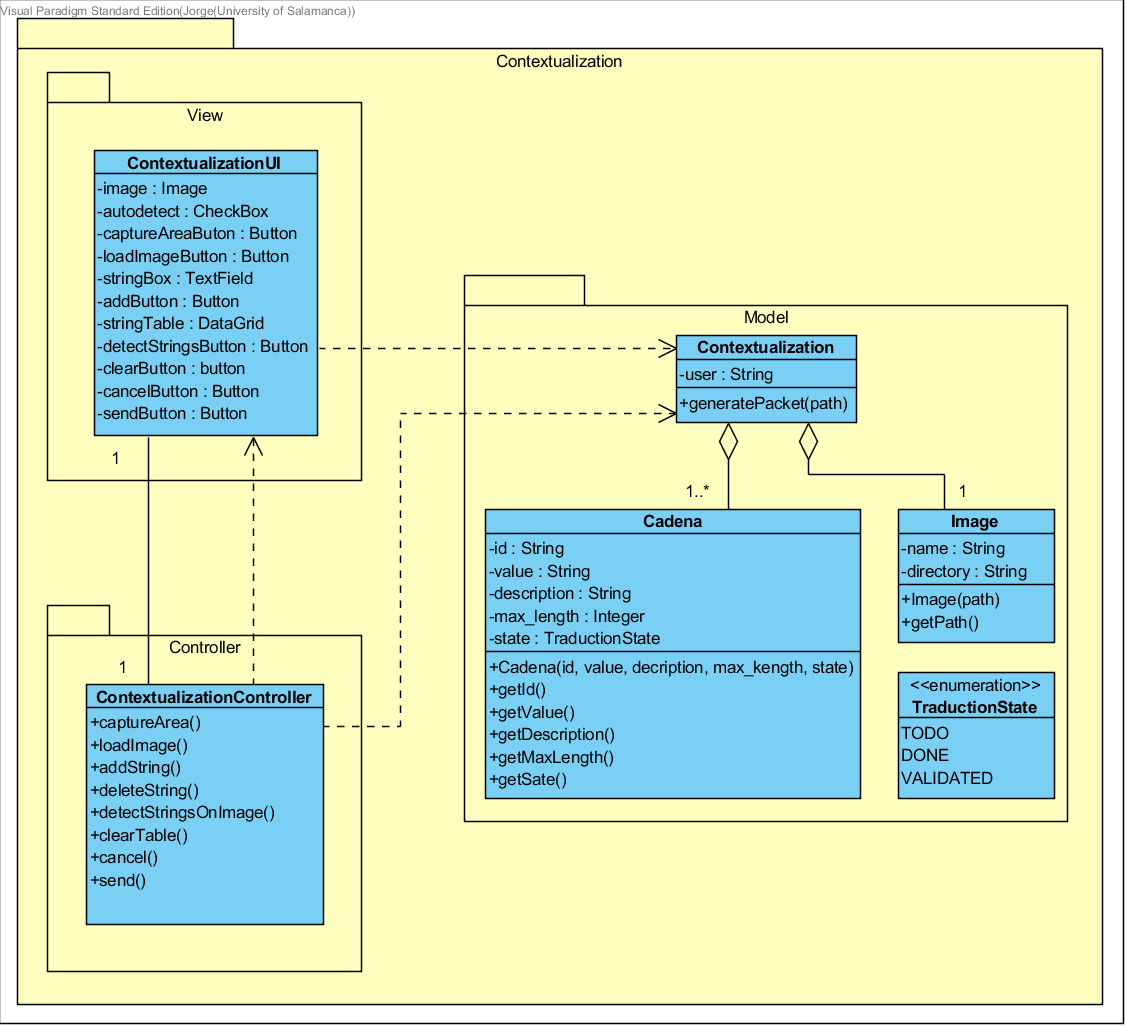
REVISION TFG



Correcciones diagrama UML de clases:

1. Vista ContextualizationUi detectStringsButton, no está relacionado con el model directamente. Esta relación se hará mediante el controlador. Ui y el modelo no pueden depender directamente entre sí mismos.
2. Modelo.
   1. El nombre de la clase ha de ser ContextualizationModel.
   2. GeneratePackage es una acción, todas las acciones han de desarrollarlas el controller. El modelo ha de ser estático, sin lógica. Ejemplo, podríamos tener 3 controladores diferentes que utilicen el mismo modelo, y cada uno de ellos tenga una forma diferente de empaquetar.
   3. El Max\_length pasar a camelCase.
   4. El modelo no tendrá una lista de cadenas y una lista de imágenes, sin relacionar como muestra el diagrama. De hecho será una imagen con una lista de cadenas. Así que la imagen pertenecerá directamente al modelo. Y el agregado de las cadenas está directamente relacionado así con la imagen del modelo. Si con vistas a futuro se quieren realizar varios modelos simultáneos se manejará una lista de ContextualizationModels.
   5. El modelo será rellenado/construido por el controller. Falta su constructor.
   6. El modelo le faltan funciones públicas con las que el controller podrá manejarlo, del tipo addNewString/deleteString, setImagePath.
   7. Le falta métodos públicos con lo que el controlador alimentará al vista del tipo getStringList, getImagePath.
   8. Al modelo le falta una variable privada de tipo lista que almacenará el agregado de cadenas.
   9. A la clase cadena le añadiremos una propiedad booleana “selected” relacionada con el checkbox asociado en la vista. Cuando revisemos las Q\_Properties de Qt se verá más claro.
3. Controlador.
   1. En principio todos los métodos son privados o protected. Es decir, no están pensados para que nadie los utilice más que el controlador en sí mismo.
   2. Le falta una variable privada de tipo conetxtualizationModel.
   3. Le falta el constructor.
   4. Le añadiremos una función validateModel. La cual validará el contenido del modelo. La iremos rellenando según las necesidades q se detecten previas al envío

***Básicamente el controller es el director de todo esto. Él creará el modelo y la vista, e interactuará entre ellos aportándoles la lógica necesaria a sus acciones y a sus relaciones.***