Spotagram
Final Presentation

Team37 leiyu miaojunl dawang
11 Dec 2015

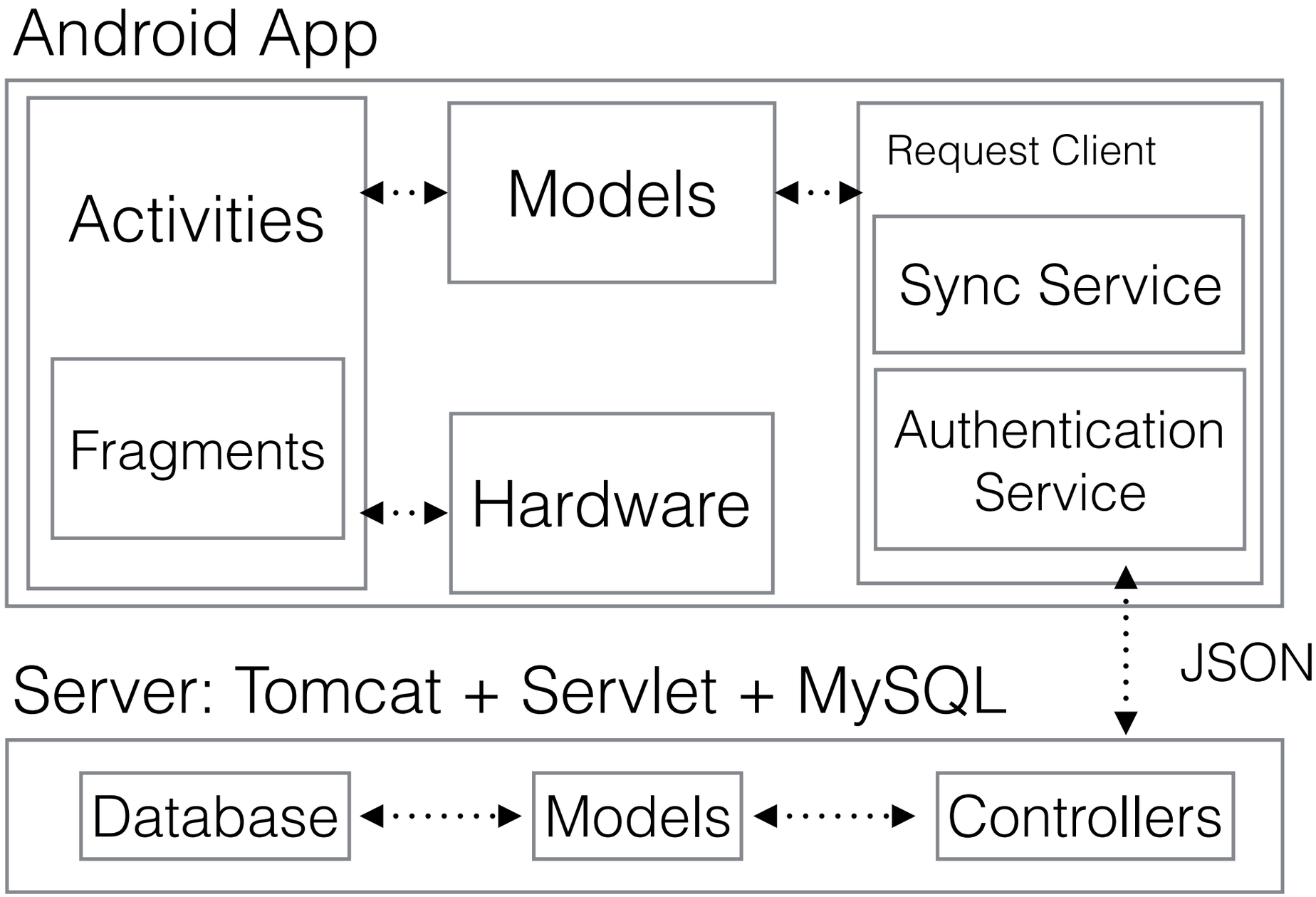
Project Idea

- **Location** Based Social App

- Be yourself! **No** follow/ unfollow relationship. You can be friends with anyone visiting the same place

- Similar to Instagram, instead of sharing photos, we **share locations** (with words/images)

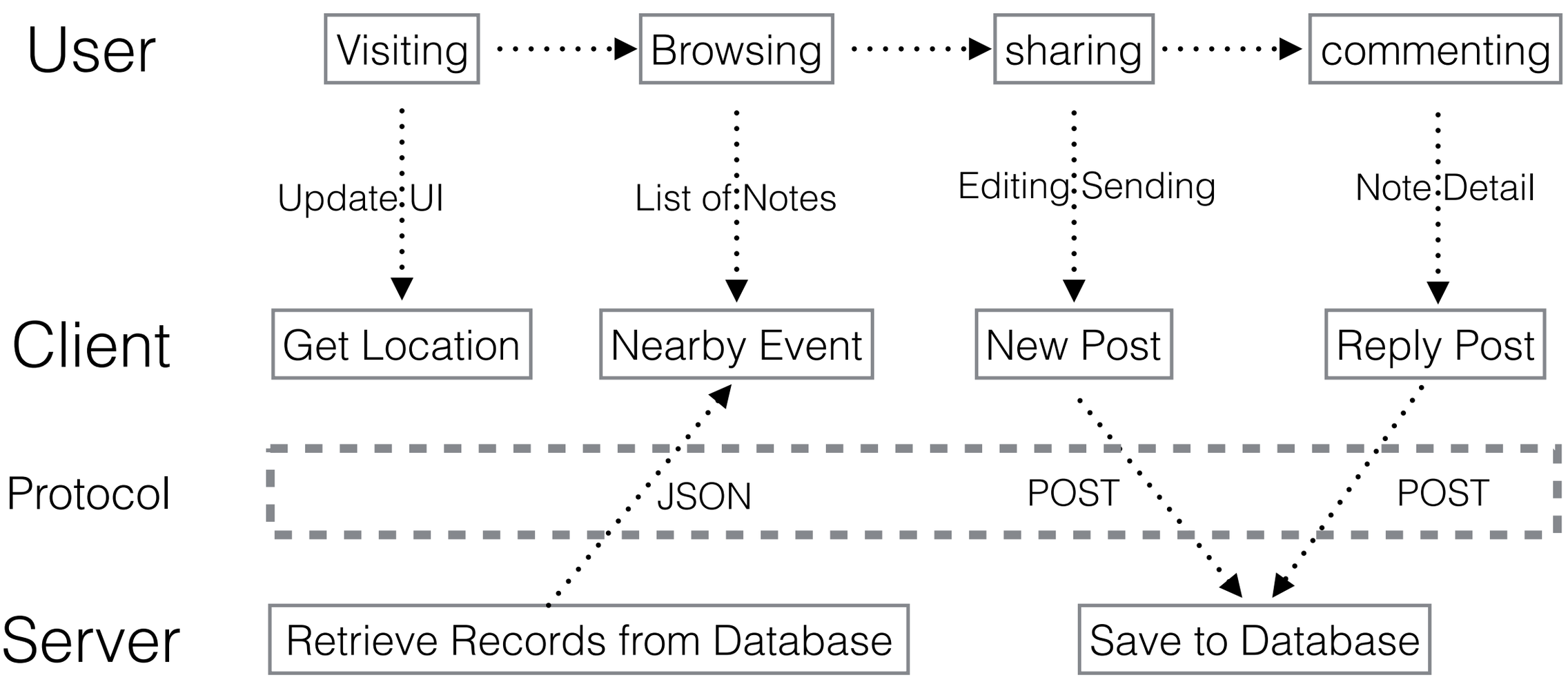
- See what's **HAPPENING** nearby on the map



Architecture

Overview of Client/Server Design
Communicate with RESTful API/JSON

User Scenario



Demo Video

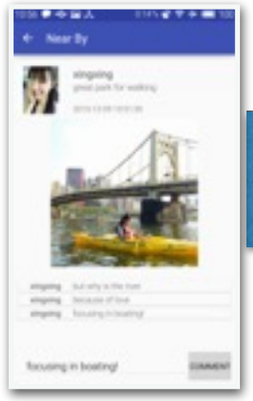
Register/Login Page

sent

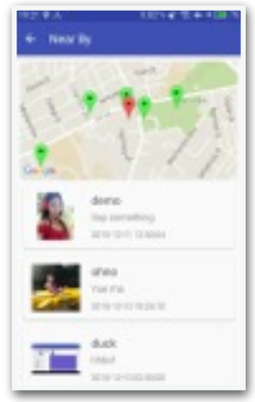
New Post Page

Map Page

Nearby Page



Detail Page



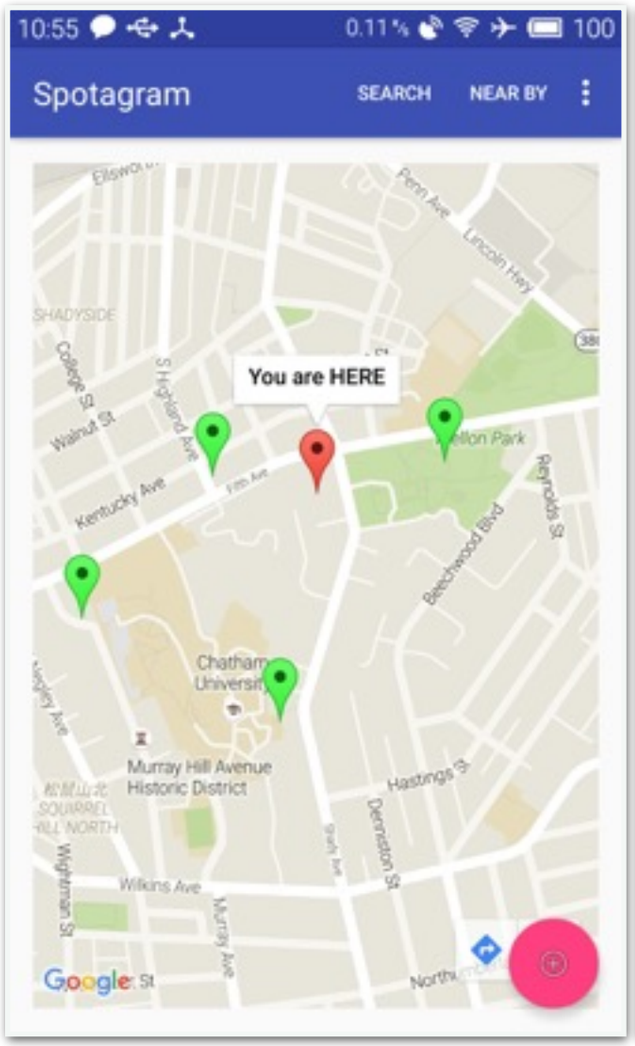
Settings Page

My Note Page

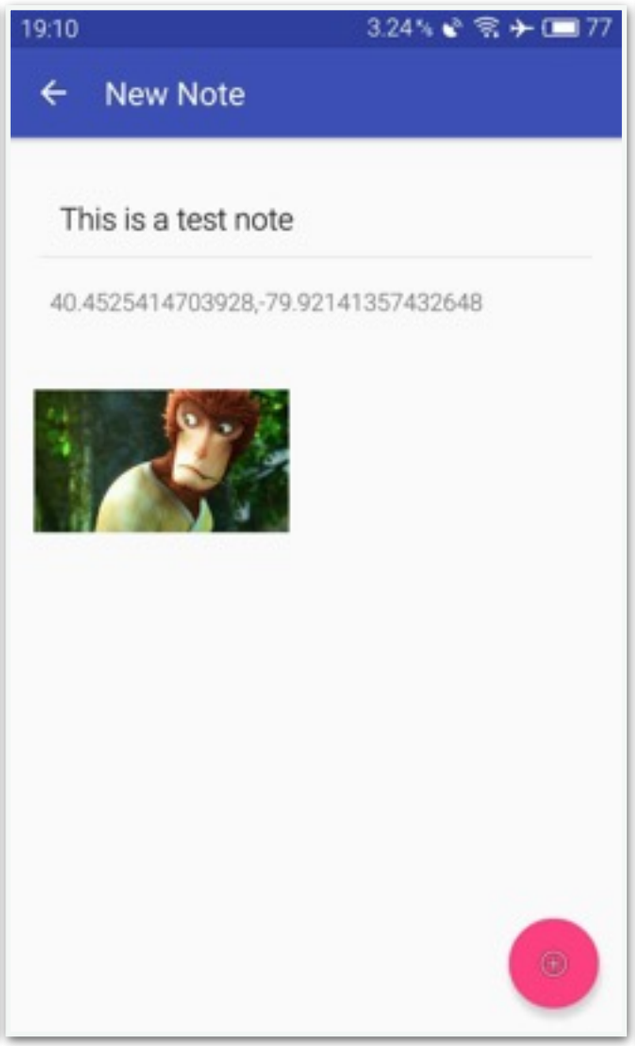
Presentation Tier

Page Flow at a glance

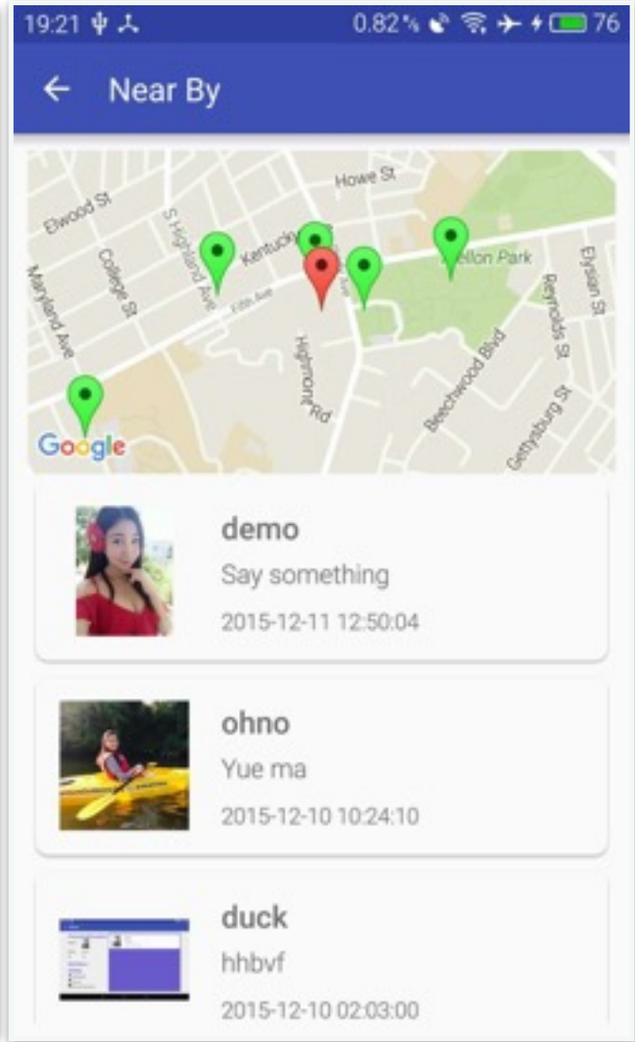
Not all interactions are shown in the graph



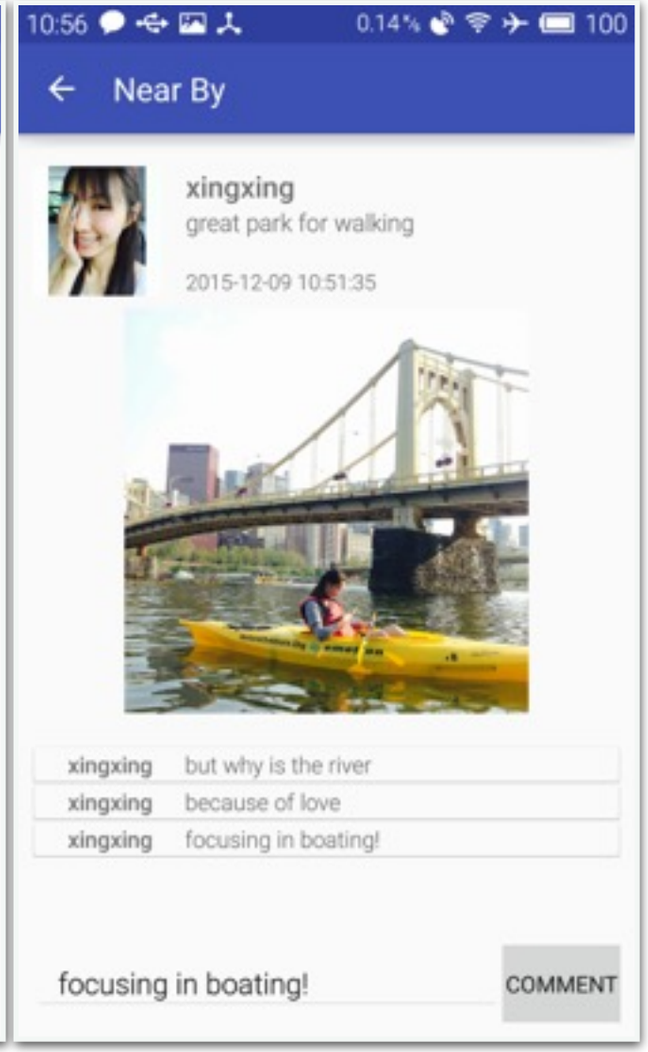
Main Page



Post New Note



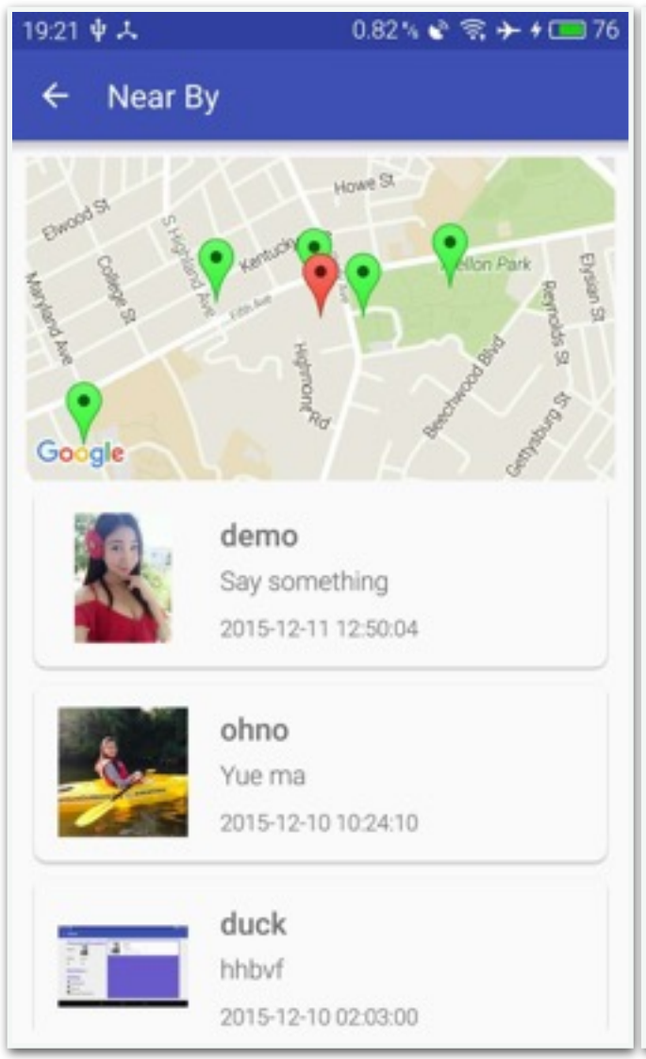
Nearby



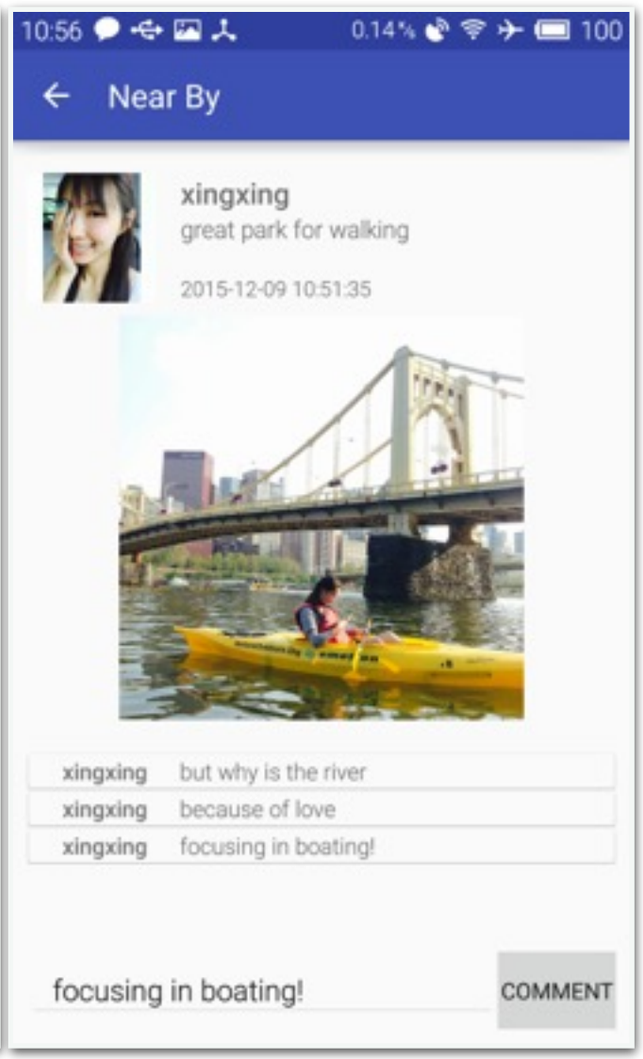
Note Detail

Presentation Tier

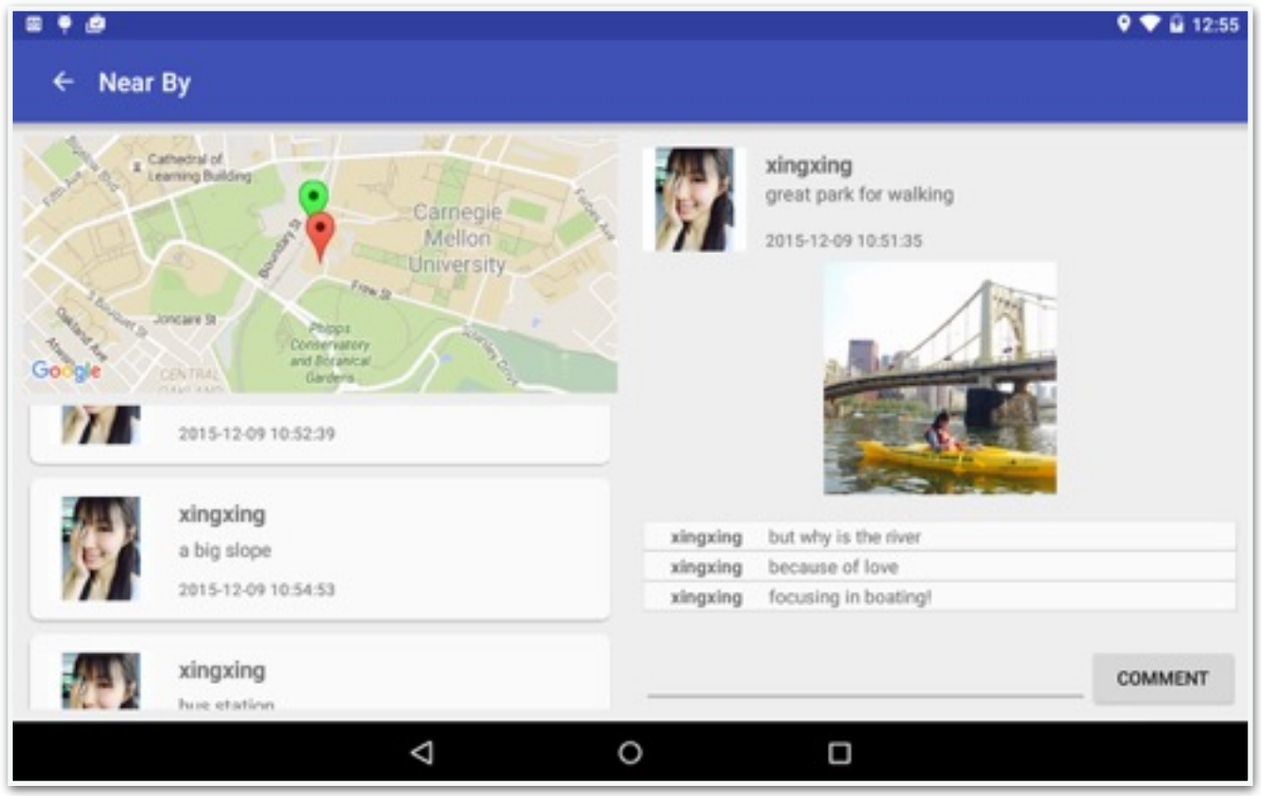
Not all pages are shown here
Introduce the main features



Nearby



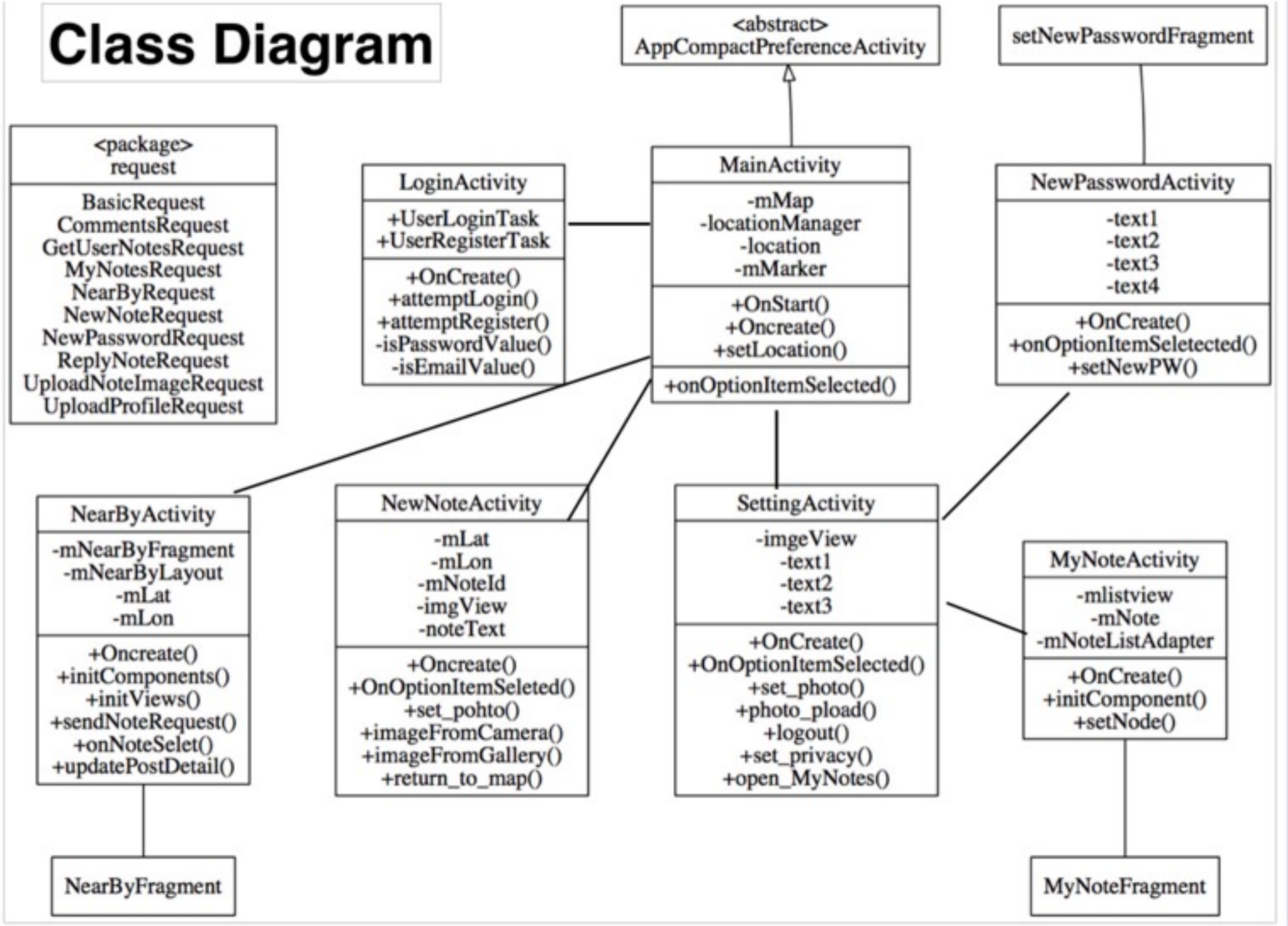
Note Detail



Nearby + Detail Page

Presentation Tier

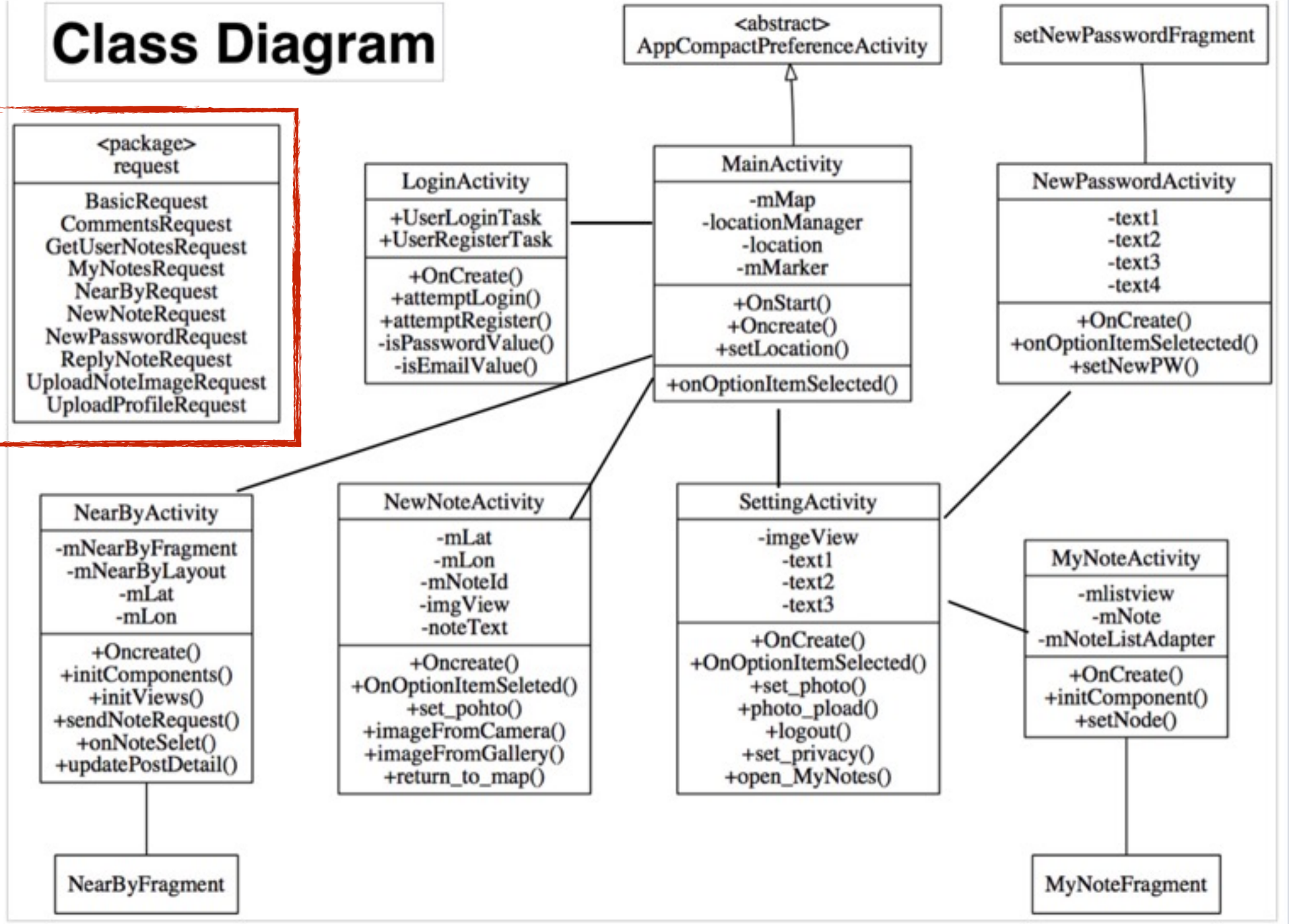
Not all pages are shown here
Introduce the main features



Application Tier

Models: getter/setter and other basic validation method

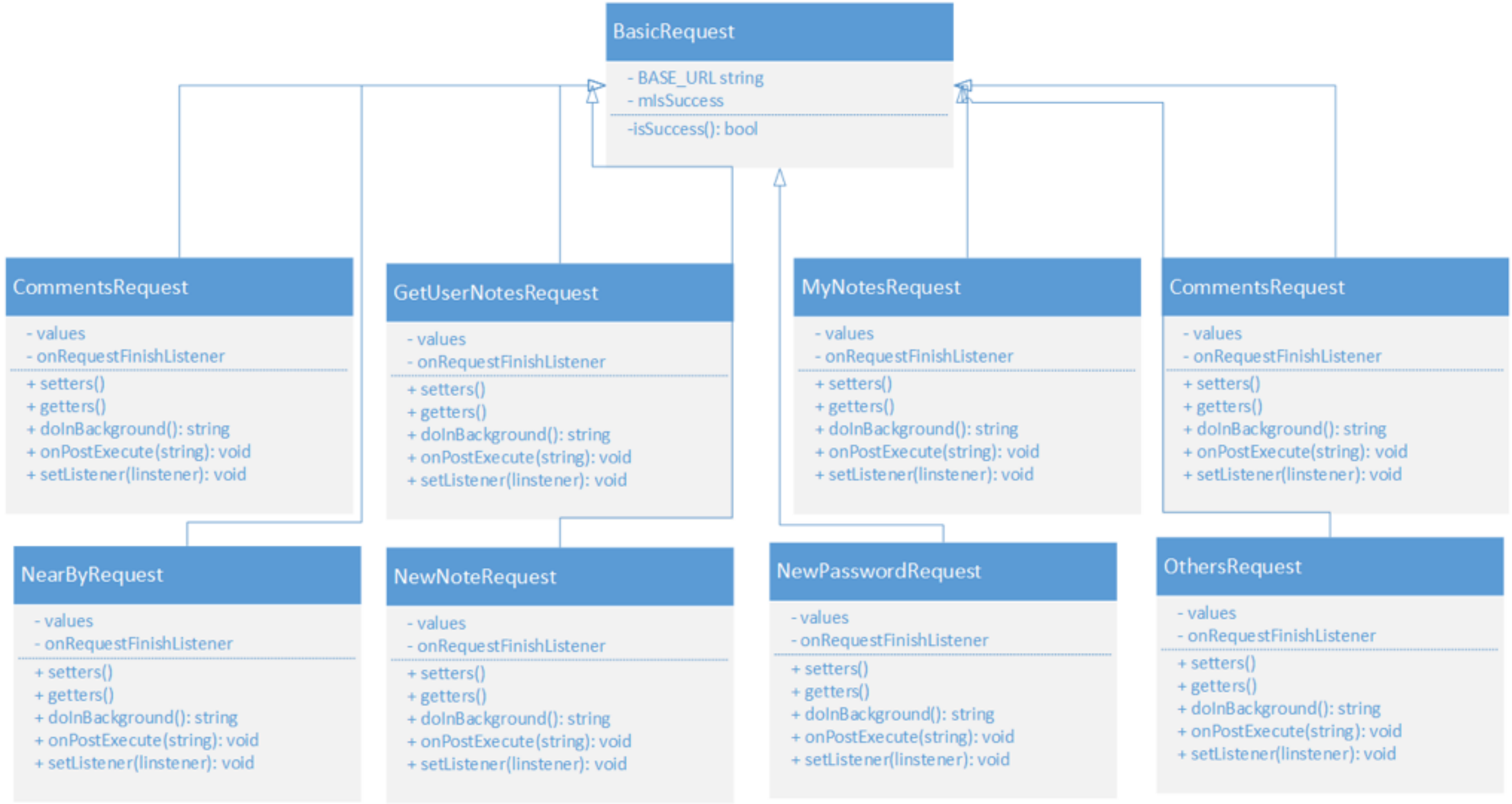
Activities: methods for starting new intent or responding button clicking



Application Tier

Models: getter/setter and other basic validation method

Activities: methods for starting new intent or responding button clicking



Content Provider

Client: SharedPreference

Fetch latest data from server using

Protocol of Registration

Name	Info
Description	In this protocol, the client send username, email and password to server to register.
Type	POST
Notice	The client should check the correctness of request type

Format of Request

Key	Type	Value
user	String	The username
email	String	User email
pwd	string	The password of user

Format of Response

Key	Type	Value
result	int	The result code about the request

Format of Result code

Key	Value	Description
RESULT_OK	0	Log in success
RESULT_USER_ERR	-1	User already existed
RESULT_EMAIL_ERR	-2	Email already existed

Protocol of getting notes

Name	Info
Description	In this protocol, the client send request to get the notes in specific area.
Type	POST
Notice	

Format of Request

Key	Type	Value
loc_longitude	float	Longitude of the location
loc_latitude	float	Latitude of the location
radius_km	int	The range of the area to get the notes
max_note	int	The maximum number of notes to get
start_note	int	The start index of notes to get (usually is 0)

Format of Response

Key	Type	Value
result	int	The result code about the request
note_lists	JSON	The notes returned

Format of Result code

Key	Value	Description
RESULT_OK	0	Log in success
RESULT_ERR	-3	Undefined

Format of JSON note

Key	Type	Description
user_id	int	Id
user_name	string	name
loc_longitude	float	Longitude of the location
loc_latitude	float	Latitude of the location
info	String	The information of note
date	date	undefined

Protocol of getting comments from a note

Name	Info
Description	In this protocol, the client get comments related to one note
Type	POST
Notice	**null**

Format of Request

Key	Type	Value
note_id	int	Note id

Format of Response

Key	Type	Value
result	int	The result code about the request

Format of Result code

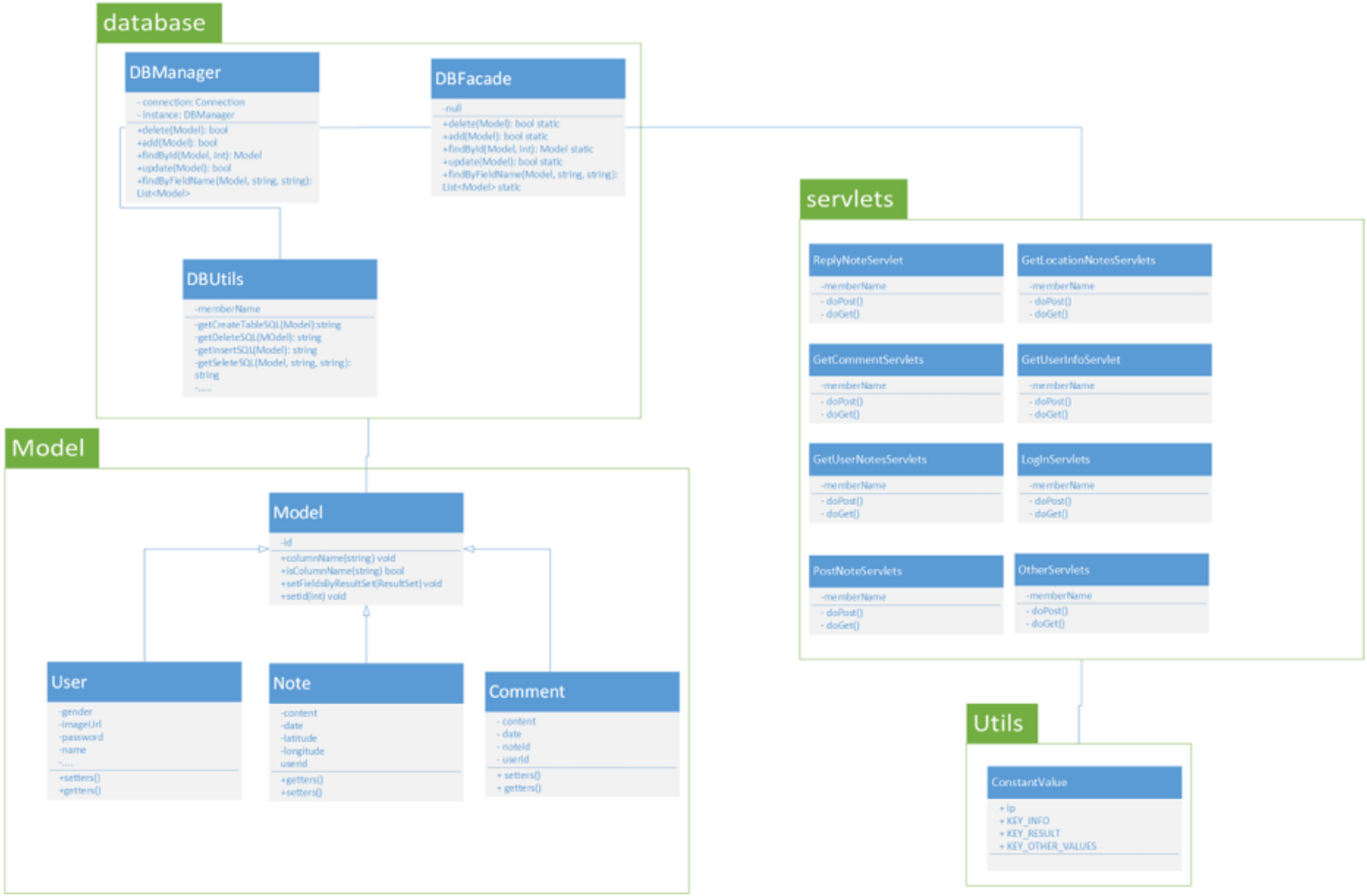
Key	Value	Description
RESULT_OK	0	Log in success
RESULT_ERR	-3	Undefined

Format of JSON note

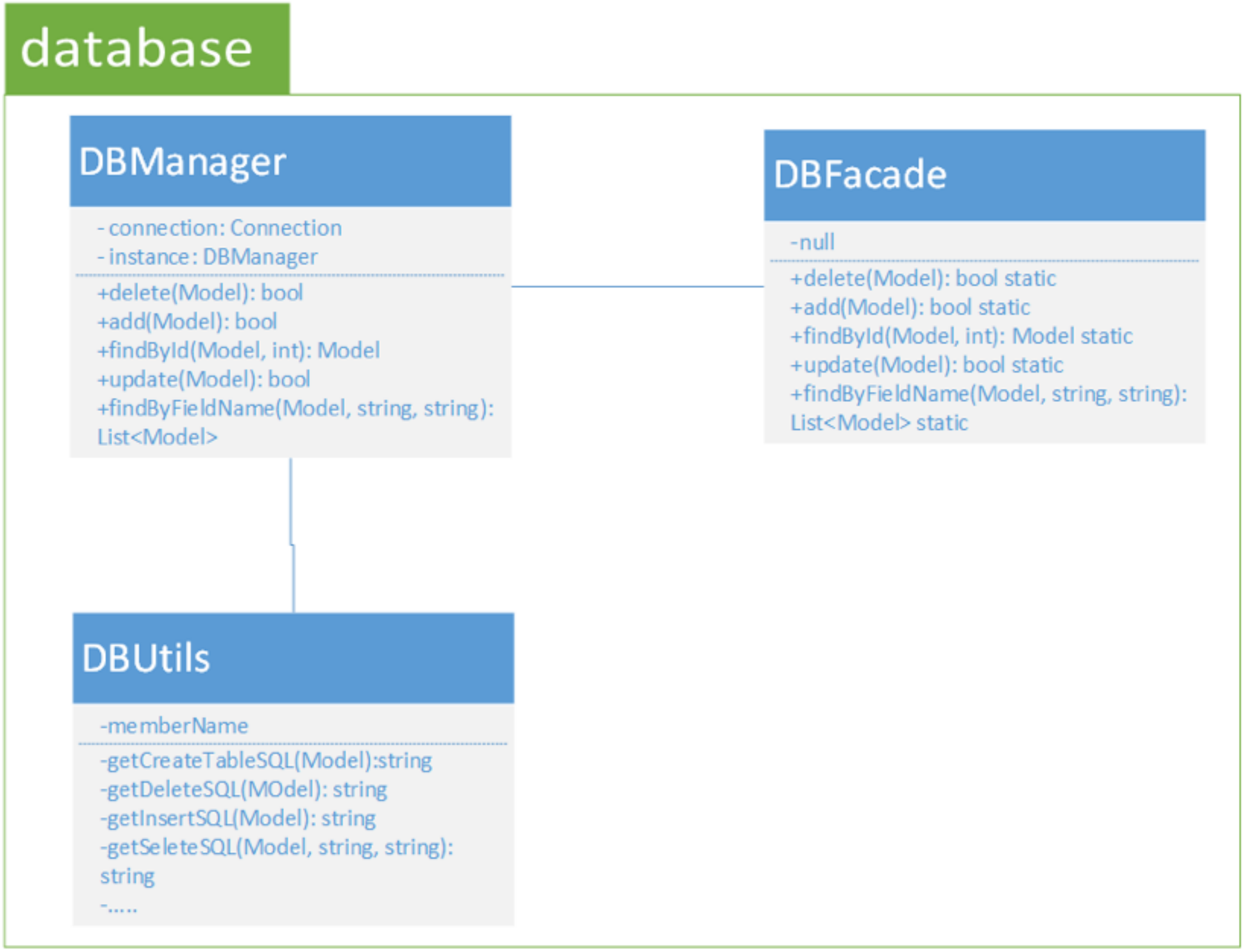
Key	Type	Description
comment_id	int	Comment id
user_id	string	User id
user_name	float	User name
content	String	The information of note
date	datetime	undefined

Protocol

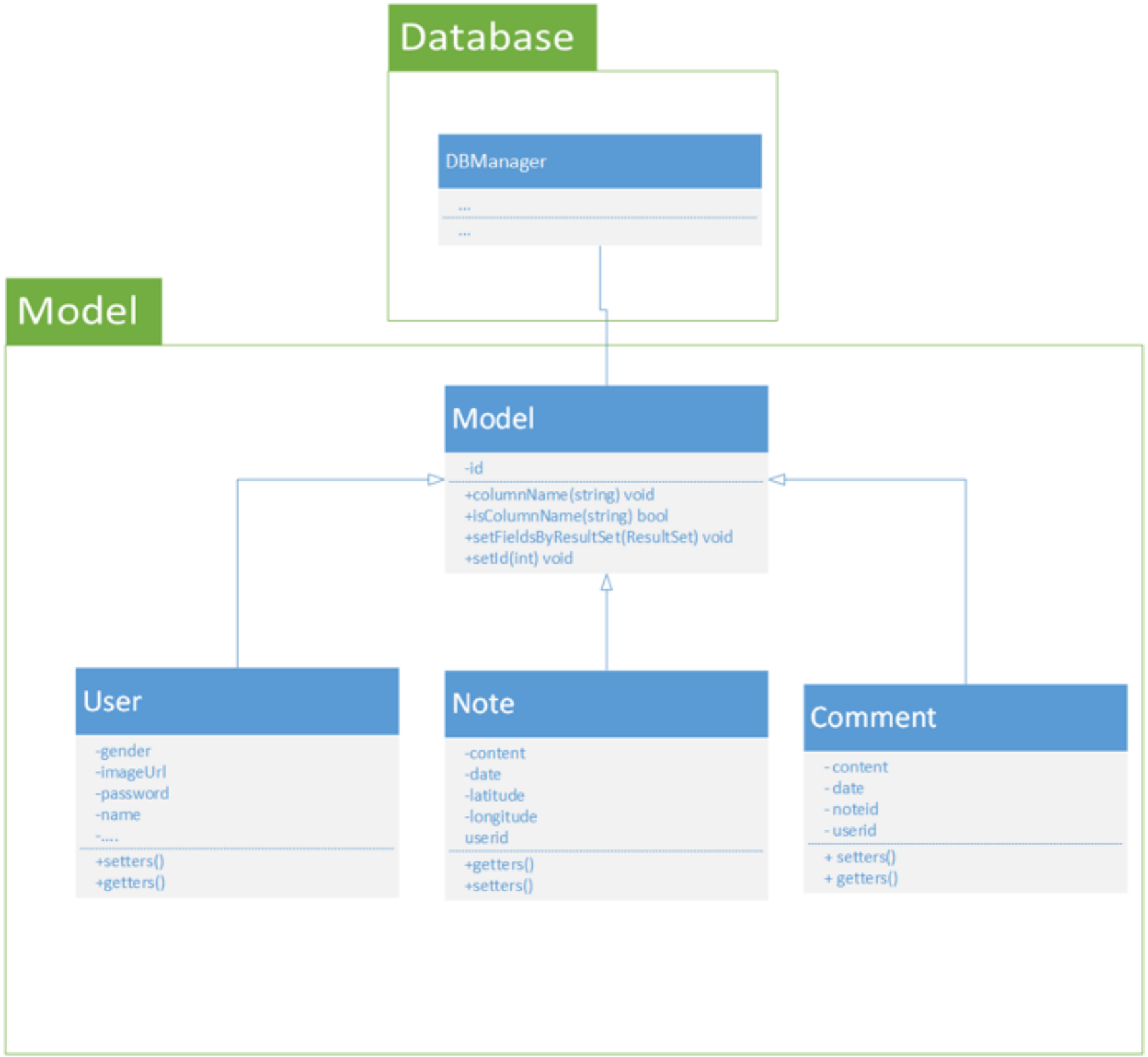
For the communication between client & server
Not all are shown


database
DBManager
- connection: Connection
- instance: DBManager
+delete(Model): bool
+add(Model): bool
+findById(Model, int): Model
+update(Model): bool
+findByFieldName(Model, string, string): List<Model>
DBFacade
-null
+delete(Model): bool static
+add(Model): bool static
+findById(Model, int): Model static
+update(Model): bool static
+findByFieldName(Model, string, string): List<Model> static
DBUtils
-memberName
-getCreateTableSQL(Model):string
-getDeleteSQL(MOdel): string
-getInsertSQL(Model): string
-getSeleteSQL(Model, string, string): string
Model
Model
-id
+columnName(string) void
+isColumnName(string) bool
+setFieldsByResultSet(ResultSet) void
+setId(int) void
User
-gender
-imageUrl
-password
-name
+setters()
+getters()
Note
-content
-date
-latitude
-longitude
userid
+getters()
+setters()
Comment
- content
- date
- noteId
- userId
+ setters()
+ getters()
servlets
ReplyNoteServlet
-memberName
- doPost()
- doGet()
GetLocationNotesServlets
-memberName
- doPost()
- doGet()
GetCommentServlets
-memberName
- doPost()
- doGet()
GetUserInfoServlet
-memberName
- doPost()
- doGet()
GetUserNotesServlets
-memberName
- doPost()
- doGet()
LogInServlets
-memberName
- doPost()
- doGet()
PostNoteServlets
-memberName
- doPost()
- doGet()
OtherServlets
-memberName
- doPost()
- doGet()
Utils
ConstantValue
+ ip
+ KEY_INFO
+ KEY_RESULT
+ KEY_OTHER_VALUES

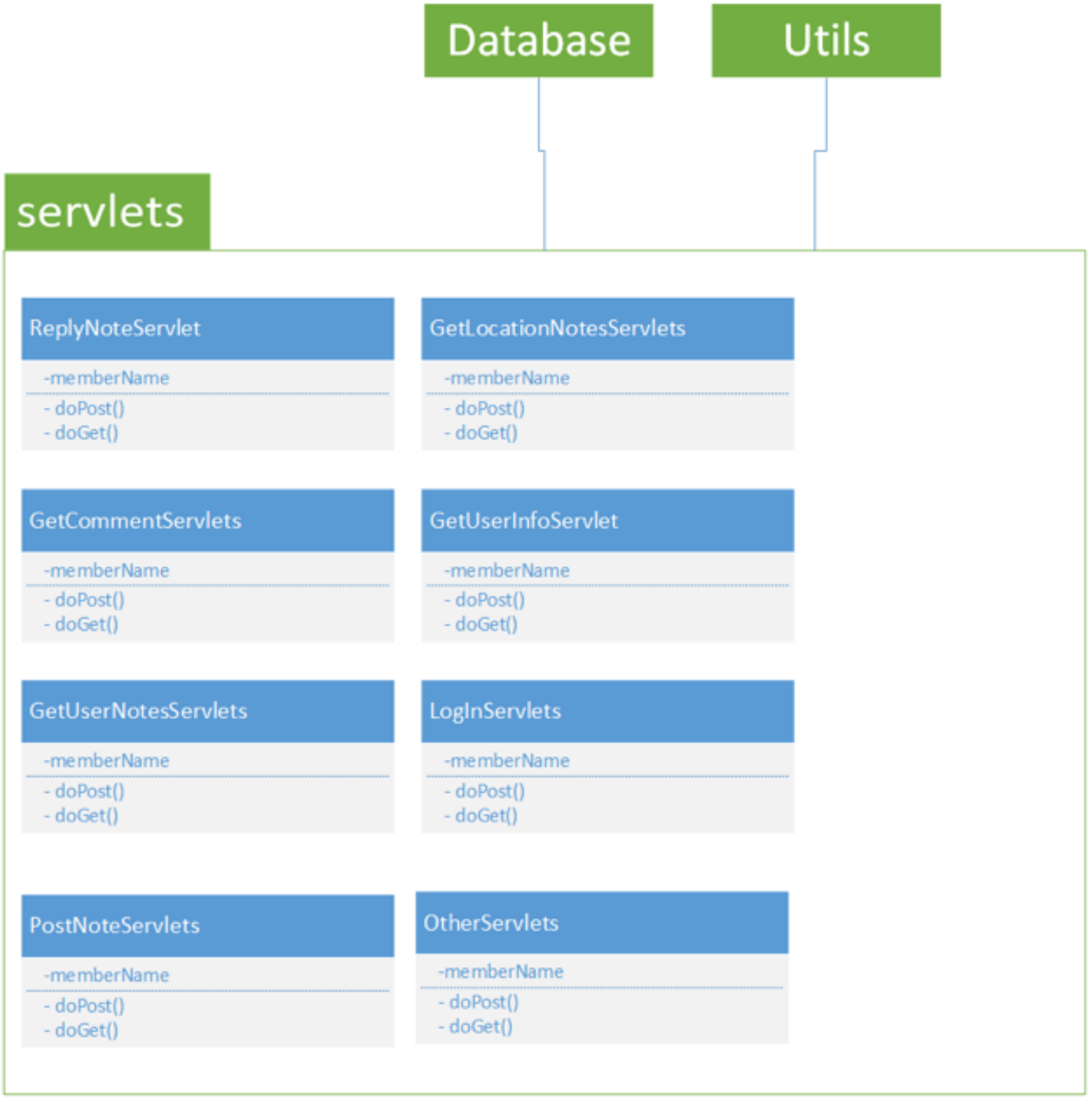
Server Tier


database
DBManager
- connection: Connection
- instance: DBManager
+delete(Model): bool
+add(Model): bool
+findById(Model, int): Model
+update(Model): bool
+findByFieldName(Model, string, string): List<Model>
DBFacade
-null
+delete(Model): bool static
+add(Model): bool static
+findById(Model, int): Model static
+update(Model): bool static
+findByFieldName(Model, string, string): List<Model> static
DBUtils
-memberName
-getCreateTableSQL(Model):string
-getDeleteSQL(MOdel): string
-getInsertSQL(Model): string
-getSeleteSQL(Model, string, string): string
-.....

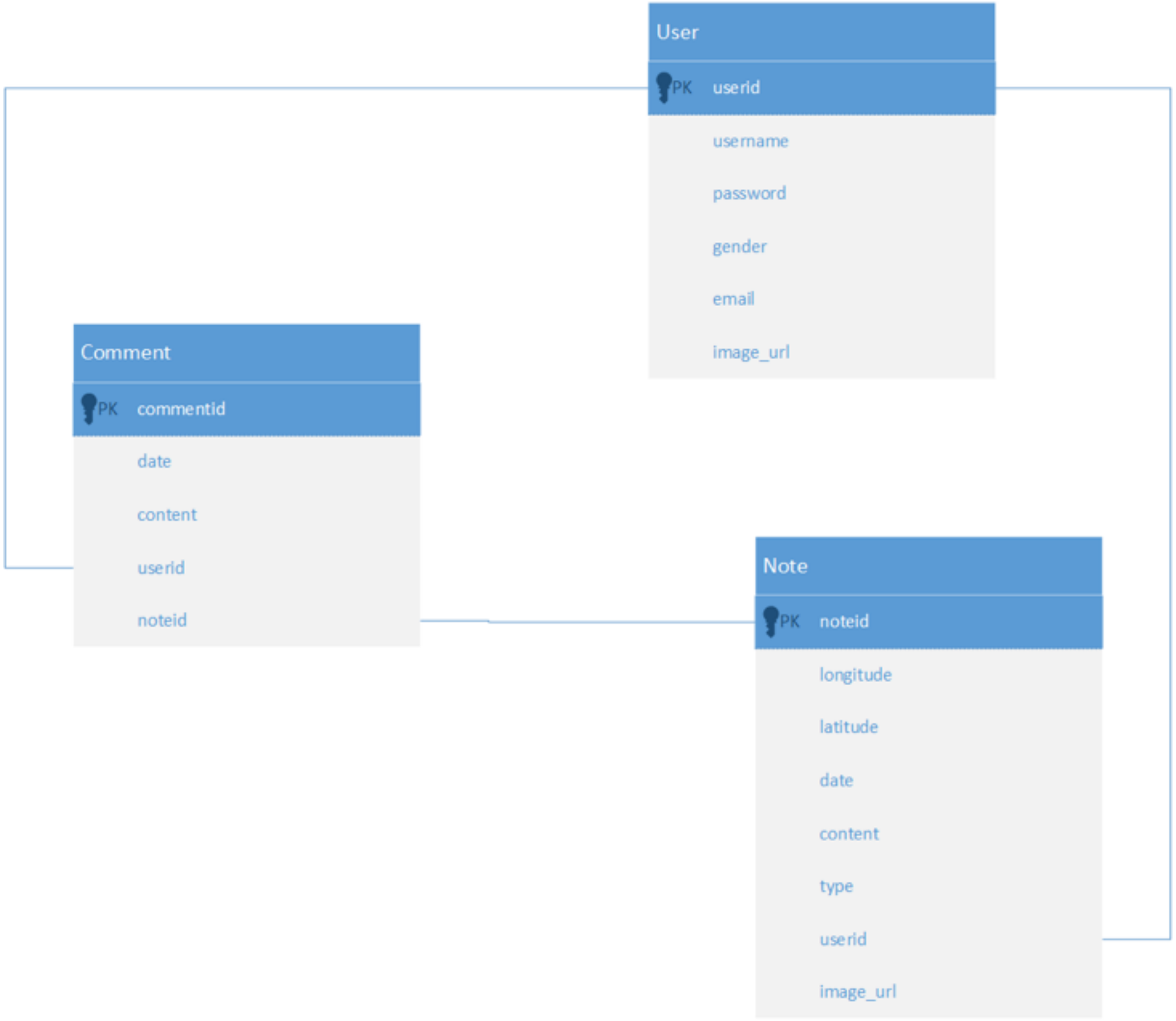
Server Tier


Database
DBManager
...
...
Model
Model
-id
+columnName(string) void
+isColumnName(string) bool
+setFieldsByResultSet(ResultSet) void
+setId(int) void
User
-gender
-imageUrl
-password
-name
-....
+setters()
+getters()
Note
-content
-date
-latitude
-longitude
userid
+getters()
+setters()
Comment
- content
- date
- noteid
- userid
+ setters()
+ getters()

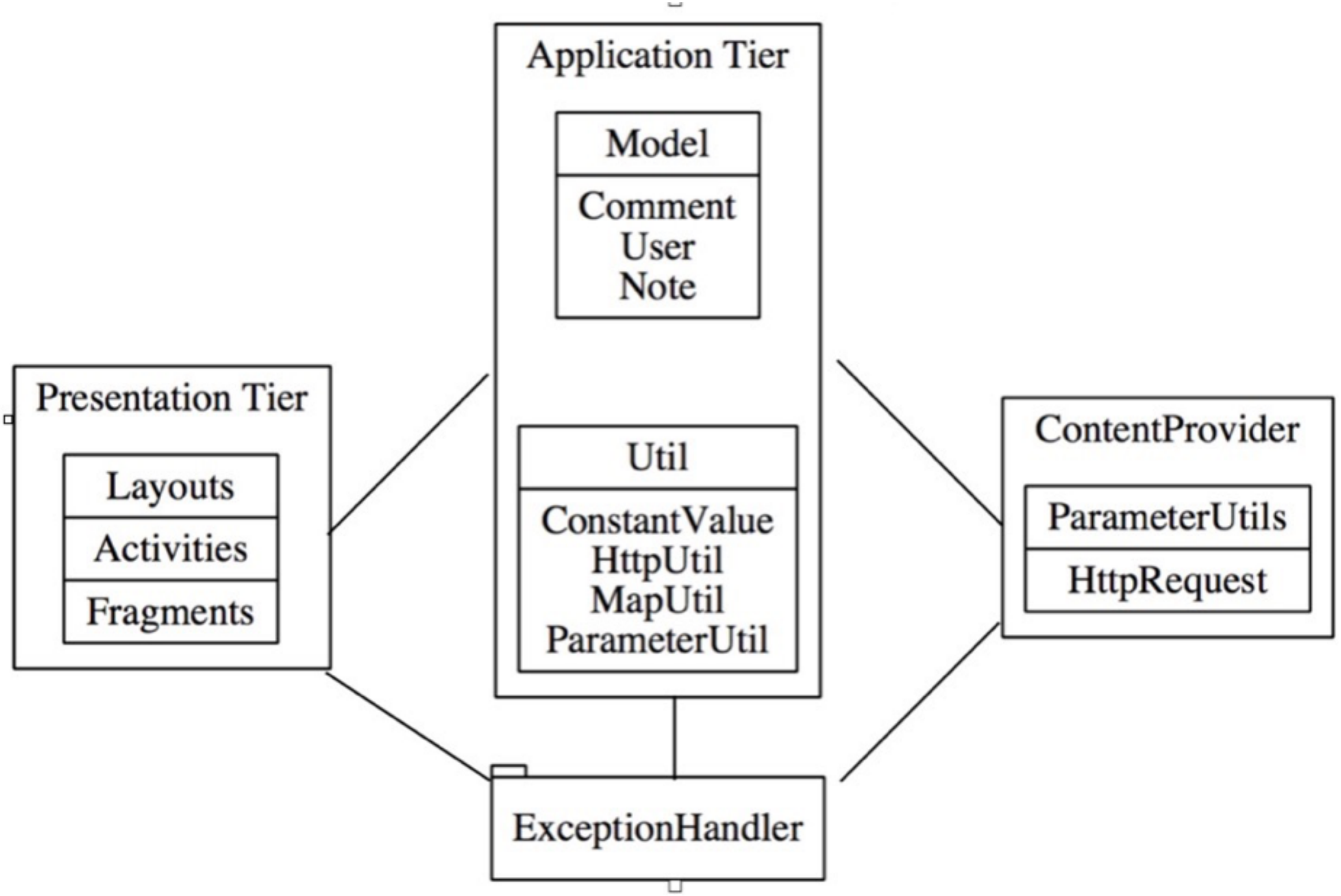
Server Tier


Database
Utils
servlets
ReplyNoteServlet
-memberName
- doPost()
- doGet()
GetLocationNotesServlets
-memberName
- doPost()
- doGet()
GetCommentServlets
-memberName
- doPost()
- doGet()
GetUserInfoServlet
-memberName
- doPost()
- doGet()
GetUserNotesServlets
-memberName
- doPost()
- doGet()
LogInServlets
-memberName
- doPost()
- doGet()
PostNoteServlets
-memberName
- doPost()
- doGet()
OtherServlets
-memberName
- doPost()
- doGet()

Server Tier



Database Schema



Interaction Between Layers

- Do the research first
- Have a plan, use checklist
- Package, Interface, Abstract Classes, OOP
- Communicate and coordinate with others
- Allow enough time
- Reuse proven code
- Test, test, test

Lessons Learnt

From Mini1 to Mini2

Thanks