A User will not have a
role if admin is true.

IDatabaseEntity

User

- firstName:String
- lastName:String
- email:String
- passwordHash:String
- address:String
- admin:boolean
- role:UserRole

+ getters
~ getPasswordHash(): String
+ getUserByEmail(email:String): User
+ createNewUser(...): User
+ getId(): String
+ serialise(): Map<String, Object>
+ deserialise(map:Map<String,Object>): void

Maintains global state and saves logged in User
across sections

<<singleton>>
MealerSystem (extends Application)

- <u>system</u>: MealerSystem
- usersession: SharedPreferences
- currentUser: User
- repository: IRepository

+ <u>getSystem</u>(): MealerSystem
+ getCurrentUser(): User
+ getRepository(): IRepository
+ onCreate(): void
+ restoreSession(): void
+ logoff(): void
+ tryLogin(email:String, password:String): boolean
- login(user:User): void
~ hashPassword(password:String): String

Generalization of a queryable permanent storage system

<<interface>>
IRepository

(T extends IRepositoryEntity)

+ *hasId*(cls:Class<T>, id:String): boolean
+ *getById*(cls:Class<T>, id:String): T
+ *set*(cls:Class<T>, entity:T): void
+ *update*(cls:Class<T>, id:String, properties:Map<String,Object>): void
+ *delete*(cls:Class<T>, entity:T): void
+ *list*(cls:Class<T>): List<T>
+ *query*(cls:Class<T>, predicate:Predicate<T>): List<T>
+ *getAutoID*(cls:Class<T>): String

current user

0..1

1

1

repository instance

1

1

user

ISerialisableEntity

role   0..1

<>
UserRole

~ user: User

+ getUser(): User

ClientRole

- cardNumber:String

+ getCardNumber(): String
+ setCardNumber(num:String): void
+ serialise(): Map<String, Object>
+ deserialise(map:Map<String,Object>): void

CookRole

- description:String
- banExpiration: long = -1
- banReason: String = ""

+ getDescription(): String
+ getBanExpiration(): long
+ getBanReason(): String
+ setRole(desc:String,exp:long,reason:String): void
+ setDescription(desc:String): void
+ setBan(exp:long,reason:String): void
+ serialise(): Map<String, Object>
+ deserialise(map:Map<String,Object>): void

<<interface>>
ISerialisableEntity

+ *serialise*(): Map<String, Object>
+ *deserialise*(map:Map<String,Object>): void

<<interface>>
IRepositoryEntity

~ <u>tablesNames</u>: Map<Class<? extends IDatabaseEntity>,String>

+ <u>getTableName</u>(cls:Class<? extends IDatabaseEntity>): String
~ *getId*(): String

Represents a Entity that can be stored in a table/document in
permanent storage
Can store other ISerialisableEntity and is responsible for calling
their serialise and deserialise methods.

CloudFirestoreRepository

- db:FirestoreFirebase

(T extends IRepositoryEntity)

+ hasId(cls:Class<T>, id:String): boolean
+ getById(cls:Class<T>, id:String): T
+ set(cls:Class<T>, entity:T): void
+ update(cls:Class<T>, id:String, properties:Map<String,Object>): void
+ delete(cls:Class<T>, entity:T): void
+ list(cls:Class<T>): List<T>
+ query(cls:Class<T>, predicate:Predicate<T>): List<T>
+ getAutoID(cls:Class<T>): String

IDatabaseEntity

Complaint

- id: String
- title: String
- content: String
- cook: User
- client: User

+ getTitle(): String
+ getContent(): String
+ getCook(): User
+ getClient(): User
+ setTitle(title:String): void
+ setContent(content:String): void
+ createNewComplaint(...): Complaint
+ getId(): String
+ serialise(): Map<String, Object>
+ deserialise(map:Map<String,Object>): void