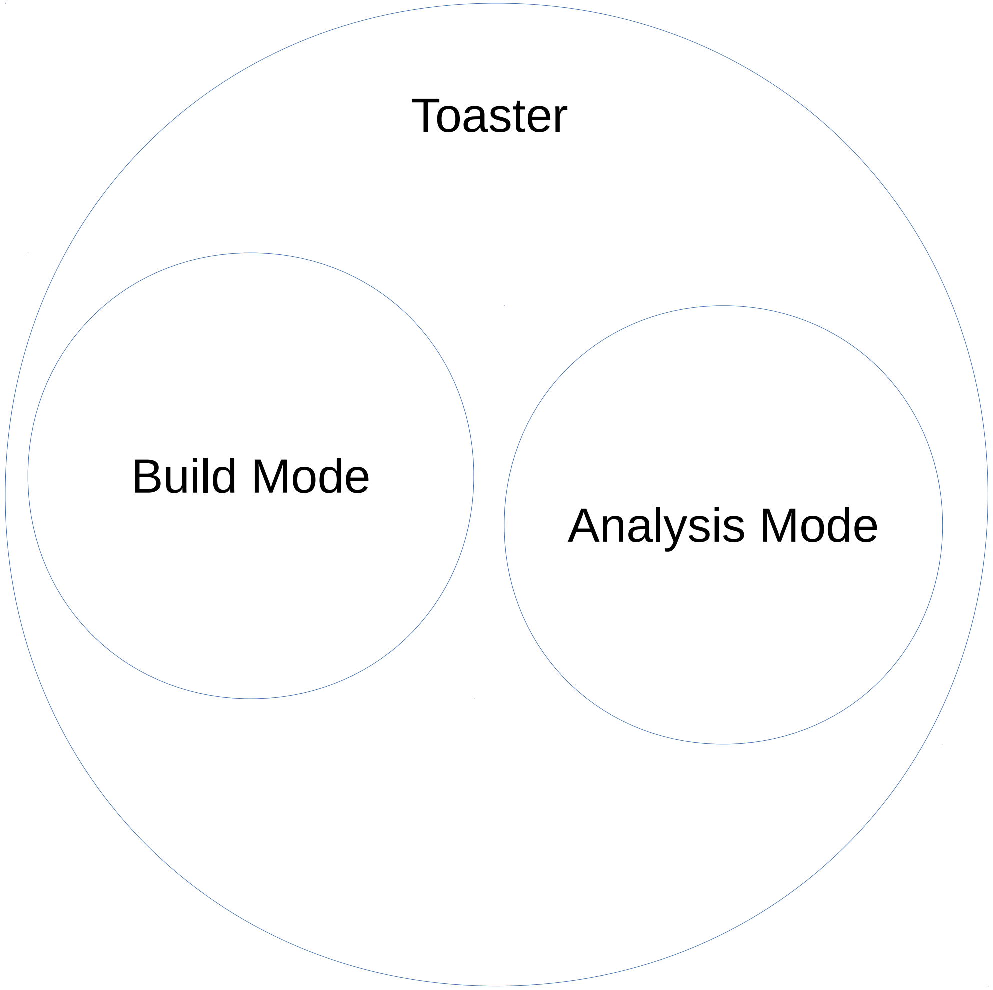
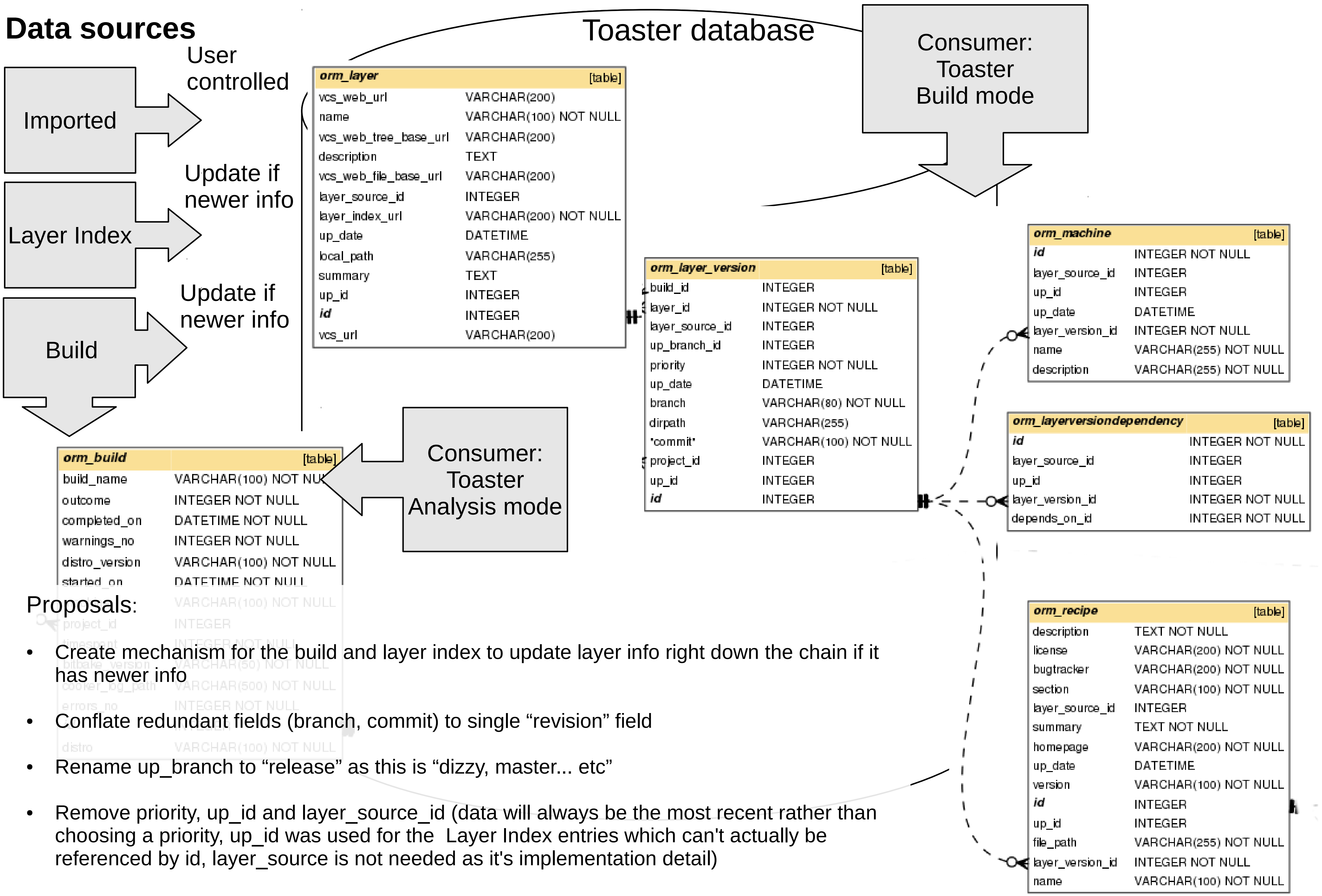
Analysis and build modes

Proposal:

- Remove any cross over from build mode and analysis mode
- Separate functionality into two distinct modules (e.g. templates/builder/ templates/analysis/ etc)



Data sources



orm_layer	[table]
vcs_web_url	VARCHAR(200)
name	VARCHAR(100) NOT NULL
vcs_web_tree_base_url	VARCHAR(200)
description	TEXT
vcs_web_file_base_url	VARCHAR(200)
layer_source_id	INTEGER
layer_index_url	VARCHAR(200) NOT NULL
up_date	DATETIME
local_path	VARCHAR(255)
summary	TEXT
up_id	INTEGER
id	INTEGER
vcs_url	VARCHAR(200)

orm_layer_version	[table]
build_id	INTEGER
layer_id	INTEGER NOT NULL
layer_source_id	INTEGER
up_branch_id	INTEGER
priority	INTEGER NOT NULL
up_date	DATETIME
branch	VARCHAR(80) NOT NULL
dirpath	VARCHAR(255)
"commit"	VARCHAR(100) NOT NULL
project_id	INTEGER
up_id	INTEGER
id	INTEGER

orm_machine	[table]
id	INTEGER NOT NULL
layer_source_id	INTEGER
up_id	INTEGER
up_date	DATETIME
layer_version_id	INTEGER NOT NULL
name	VARCHAR(255) NOT NULL
description	VARCHAR(255) NOT NULL

orm_layerversiondependency	[table]
id	INTEGER NOT NULL
layer_source_id	INTEGER
up_id	INTEGER
layer_version_id	INTEGER NOT NULL
depends_on_id	INTEGER NOT NULL

orm_build	[table]
build_name	VARCHAR(100) NOT NU
outcome	INTEGER NOT NULL
completed_on	DATETIME NOT NULL
warnings_no	INTEGER NOT NULL
distro_version	VARCHAR(100) NOT NULL
started_on	DATETIME NOT NULL
machine	VARCHAR(100) NOT NULL
project_id	INTEGER
timespent	INTEGER NOT NULL
bitbake_version	VARCHAR(50) NOT NULL
cooker_log_path	VARCHAR(500) NOT NULL
errors_no	INTEGER NOT NULL
id	INTEGER
distro	VARCHAR(100) NOT NULL

orm_recipe	[table]
description	TEXT NOT NULL
license	VARCHAR(200) NOT NULL
bugtracker	VARCHAR(200) NOT NULL
section	VARCHAR(100) NOT NULL
layer_source_id	INTEGER
summary	TEXT NOT NULL
homepage	VARCHAR(200) NOT NULL
up_date	DATETIME
version	VARCHAR(100) NOT NULL
id	INTEGER
up_id	INTEGER
file_path	VARCHAR(255) NOT NULL
layer_version_id	INTEGER NOT NULL
name	VARCHAR(100) NOT NULL

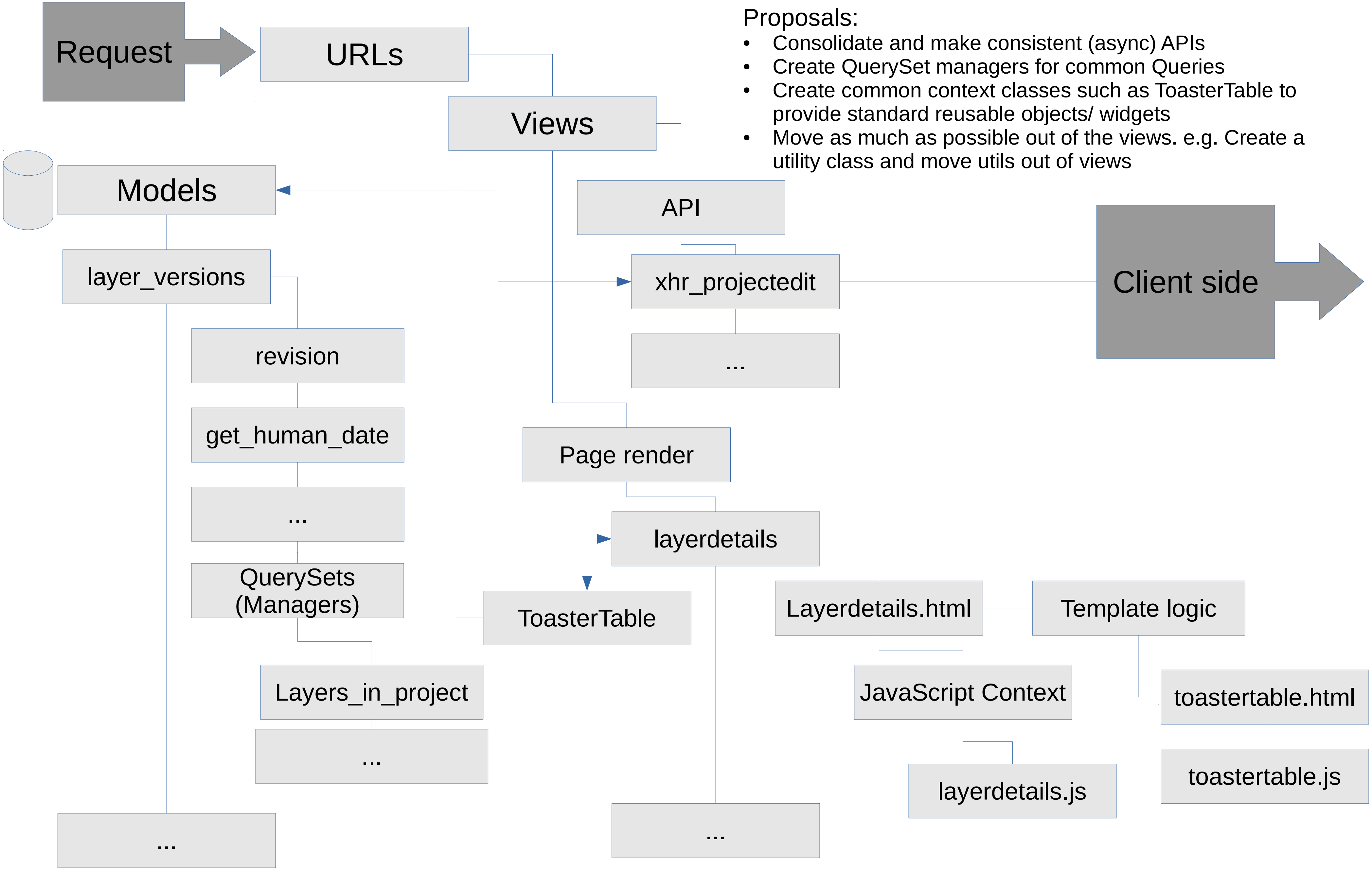
Proposals:

- Create mechanism for the build and layer index to update layer info right down the chain if it has newer info
- Conflate redundant fields (branch, commit) to single “revision” field
- Rename up_branch to “release” as this is “dizzy, master... etc”
- Remove priority, up_id and layer_source_id (data will always be the most recent rather than choosing a priority, up_id was used for the Layer Index entries which can't actually be referenced by id, layer_source is not needed as it's implementation detail)

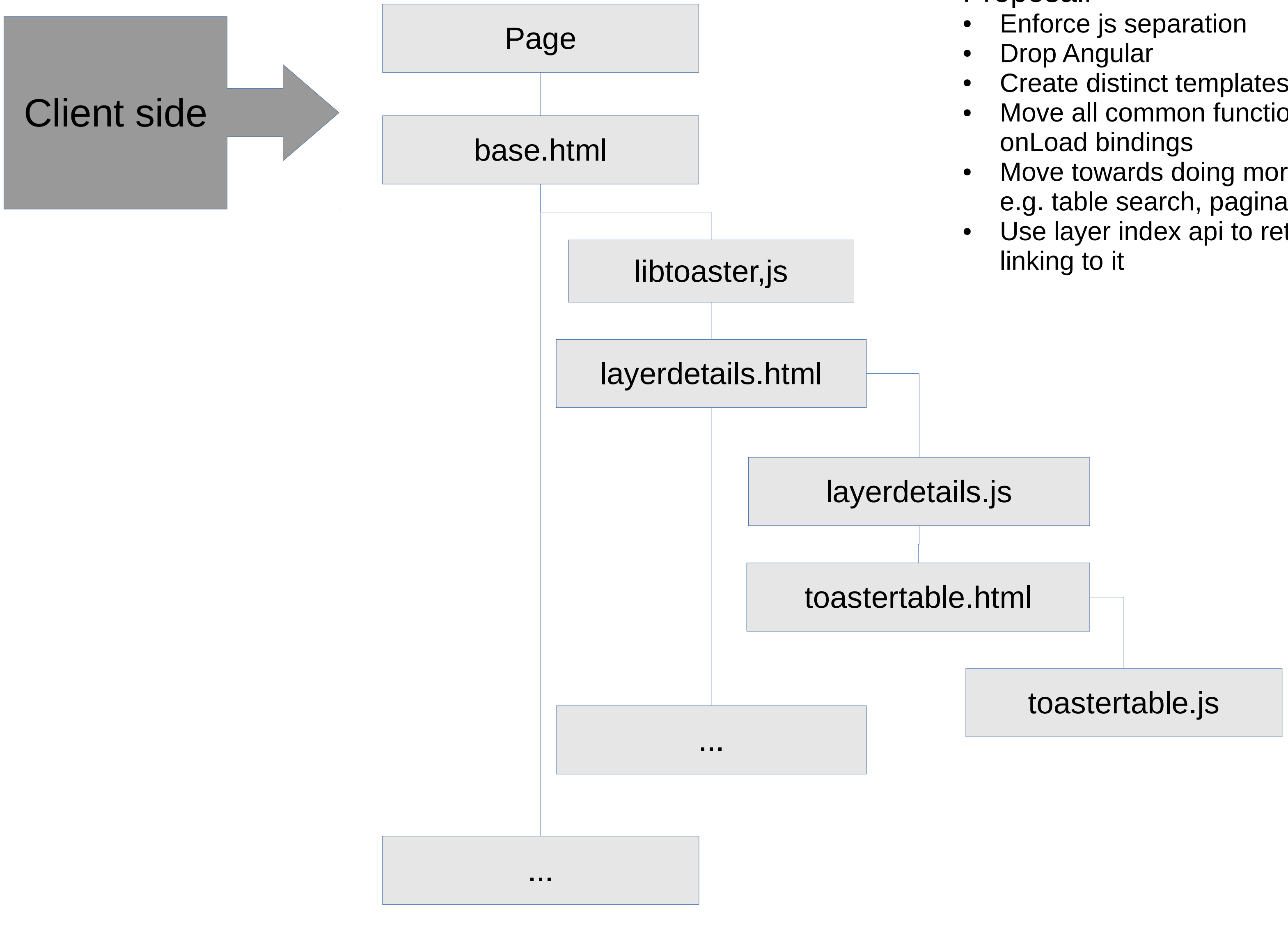
Server side - Django

Proposals:

- Consolidate and make consistent (async) APIs
- Create QuerySet managers for common Queries
- Create common context classes such as ToasterTable to provide standard reusable objects/ widgets
- Move as much as possible out of the views. e.g. Create a utility class and move utils out of views



Client Side



Proposal:

- Enforce js separation
- Drop Angular
- Create distinct templates which are /widgets
- Move all common functions to libtoaster including all onLoad bindings
- Move towards doing more functionality on the client side, e.g. table search, pagination.
- Use layer index api to return information rather than linking to it

Misc

Proposals

- Document a coding style guide for Django Templates, JavaScript, Python (bitbake's?) and HTML
- Make sure unit tests are written where appropriate
- Make use of Jshint and other code checking tools