ClipStash

Recap & Next Steps

Type Aliases

```rust
pub type AppDatabase = Database<Sqlite>;
pub type DatabasePool = sqlx::sqlite::SqlitePool;
pub type Transaction<'t> = sqlx::Transaction<'t, Sqlite>;
pub type AppDatabaseRow = sqlx::sqlite::SqliteRow;
pub type AppQueryResult = sqlx::sqlite::SqliteQueryResult;

pub struct Database<D: sqlx::Database>(sqlx::Pool<D>);
```

Implementation

```
impl Database<Sqlite> {
    pub async fn new(connection_str: &str) -> Self {…
    }
    pub fn get_pool(&self) -> &DatabasePool {…
    }
}
```

Database IDs

```rust
pub struct DbId(Uuid);
```

```
780ea179-cd47-4dfb-aabc-d7d378095bc1
```

Clip Model

```rust
pub struct Clip {
    pub(in crate::data) clip_id: String,
    pub(in crate::data) shortcode: String,
    pub(in crate::data) content: String,
    pub(in crate::data) title: Option<String>,
    pub(in crate::data) posted: NaiveDateTime,
    pub(in crate::data) expires: Option<NaiveDateTime>,
    pub(in crate::data) password: Option<String>,
    pub(in crate::data) hits: i64,
}
```

Model Clip to Domain Clip

```rust
impl TryFrom<Clip> for crate::domain::Clip {
    type Error = ClipError;
    fn try_from(clip: Clip) -> Result<Self, Self::Error> {…
    }
}
```

Queries

```rust
pub async fn get_clip<M: Into<model::GetClip>>(
    model: M,
    pool: &DatabasePool,
) -> Result<model::Clip> {

pub async fn new_clip<M: Into<model::NewClip>>(
    model: M,
    pool: &DatabasePool,
) -> Result<model::Clip> {
```

Query Models

```rust
pub struct GetClip {
    pub(in crate::data) shortcode: String,
}

pub struct UpdateClip {
    pub(in crate::data) shortcode: String,
    pub(in crate::data) content: String,
    pub(in crate::data) title: Option<String>,
    pub(in crate::data) expires: Option<i64>,
    pub(in crate::data) password: Option<String>,
}
```

Terminology

Domain
Business Rules

Model
Data Layer

Ask
Service Layer