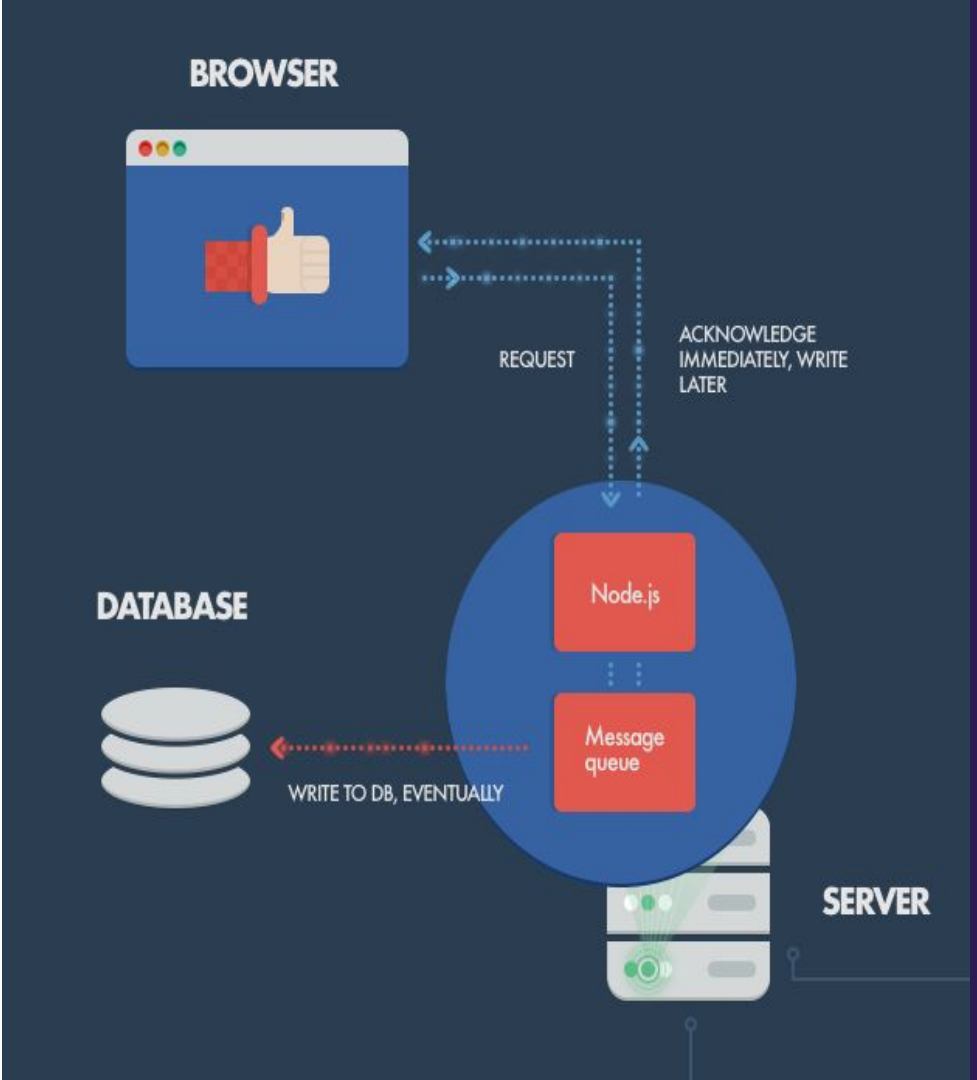
Node / Base de datos

DEV.F:
DESARROLLAMOS(PERSONAS);



Backend

Para hacer un backend casi siempre se necesita persistencia de datos, para ello necesitamos una base de datos.

Existen diferentes maneras de lograr una conexión a la base de datos.

DEV.F::

```
#!/usr/bin/env node

// $ npm install pg

const { Client } = require('pg');
const connection = require('./connection.json');
const client = new Client(connection);

client.connect();

const query = `SELECT
  ingredient.*, item.name AS item_name, item.type AS item_type
FROM
  ingredient
LEFT JOIN
  item ON item.id = ingredient.item_id
WHERE
```

Database driver

En este nivel, estará escribiendo cadenas SQL sin procesar y entregándolas a una base de datos y recibiendo una respuesta de la base de datos.

DEV.F::

```
  .select([
    '*',
    client.ref('item.name').as('item_name'),
    client.ref('item.type').as('item_type'),
  ])
  .from('ingredient')
  .leftJoin('item', 'item.id', 'ingredient.item_id')
  .where('dish_id', '=', 1)
  .debug()
  .then(rows => {
    console.log('Ingredients:');
    for (let row of rows) {
      console.log(`${row.item_name}: ${row.quantity} ${row.unit}`);
    }

    client.destroy();
});
```

Query builder

Este es el nivel intermedio entre el uso del módulo de controlador de base de datos más simple y un ORM completo. El módulo más notable que opera en esta capa es Knex.

Este módulo puede generar consultas para algunos dialectos SQL diferentes. Este módulo depende de una de las bibliotecas antes mencionadas; necesitará instalar las que planea usar con Knex.

Tienen Migrations que permiten llevar un control en los cambios en tus tablas

DEV.F::

```
#!/usr/bin/env node

// $ npm install sequelize pg

const Sequelize = require('sequelize');
const connection = require('./connection.json');
const DISABLE_SEQUELIZE_DEFAULTS = {
  timestamps: false,
  freezeTableName: true,
};

const { DataTypes } = Sequelize;
const sequelize = new Sequelize({
  database: connection.database,
  username: connection.user,
  host: connection.host,
  port: connection.port,
```

ORM (Object Relational Mapping)

Este es el nivel más alto de abstracción que vamos a considerar. Cuando trabajamos con ORM, normalmente necesitamos hacer mucha más configuración antes de tiempo.

El ORM es un modelo de programación que transforma las tablas de una base de datos en entidades para simplificar enormemente la tarea del programador.

¿Cómo conecto mi lenguaje de programación con mi base de datos?

Level	Name	Ventajas	Desventajas	Bibliotecas
Bajo	Database driver	Tu tienes el control de todo, son más rápidos.	Es dificil de escalar, toda la seguridad depende de ti	pg, mysql
Medio	Query builder	Son más flexibles	Solucionan a medias las cosas	knex
Alto	ORM	Maneja todo tipo de conexión	Son más difíciles de configurar	Sequelize, objection