Happy Go Lucky Introduction

Dirk Riehle, FAU Erlangen

ADAP C01

Happy Go Lucky Vision

Happy Go Lucky (HGL) is

- A web app to support our project based teaching

Original HGL solutions are

- Happiness index
- Standup emails
- Code tracking

Setup and Test of Happy Go Lucky

Fork happy-go-lucky to your account, e.g. friedalex

```
git clone git@github.com:friedalex/happy-go-lucky.git

cd happy-go-lucky

npm install

npm run build

npm run test

npm run generate-mockdata

npm run test

npm run dev
```

Happy Go Lucky Base Design

HGL is a web app to support our project based teaching

An admin (professor) can create courses (by semester)

Courses have an associated schedule (homework delivery dates)

Users (anyone) can create projects and add them to courses

A project can have one or more members (ADAP = 1, AMOS = 6..12)

A project is linked to exactly one GitHub repository

ADAP and AMOS

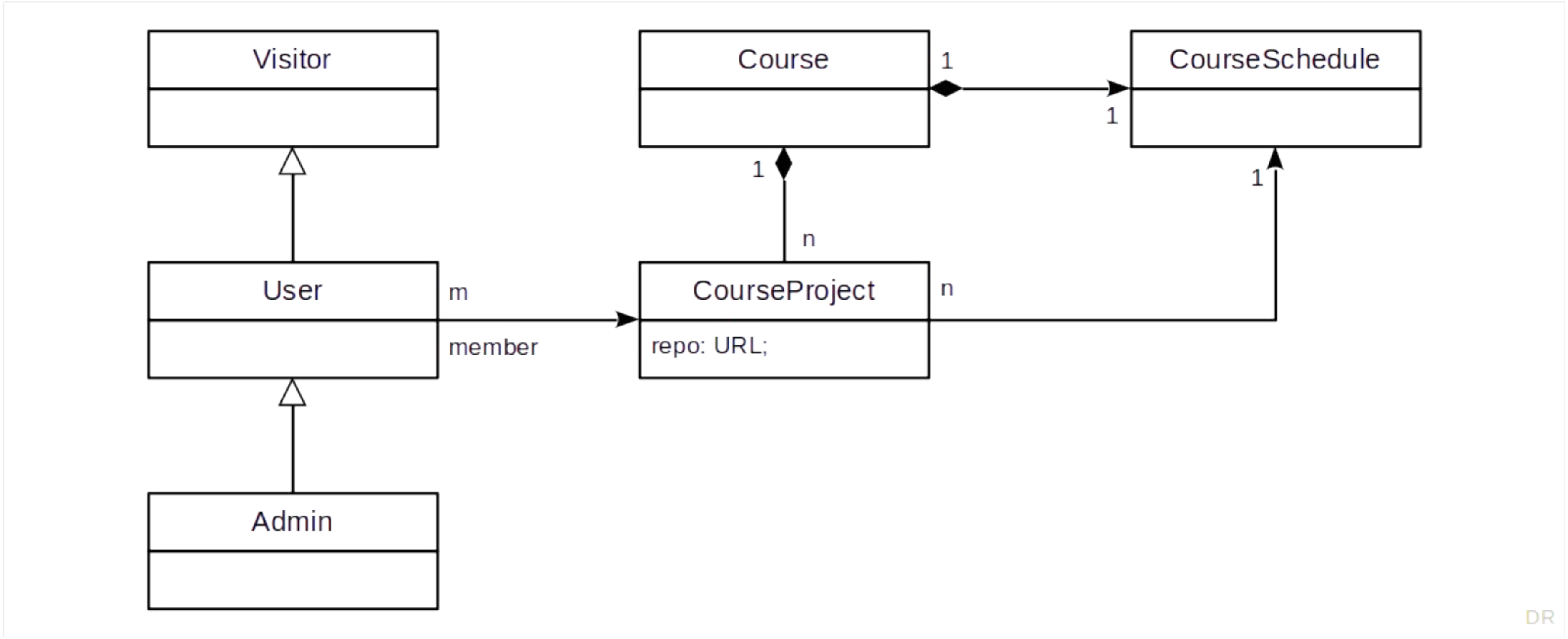
ADAP

- One course for a given term
- Each student their own project
- Students can create their project

AMOS

- One course for a given term
- Small number of set projects
- Students join existing project

Class Model (Logically)



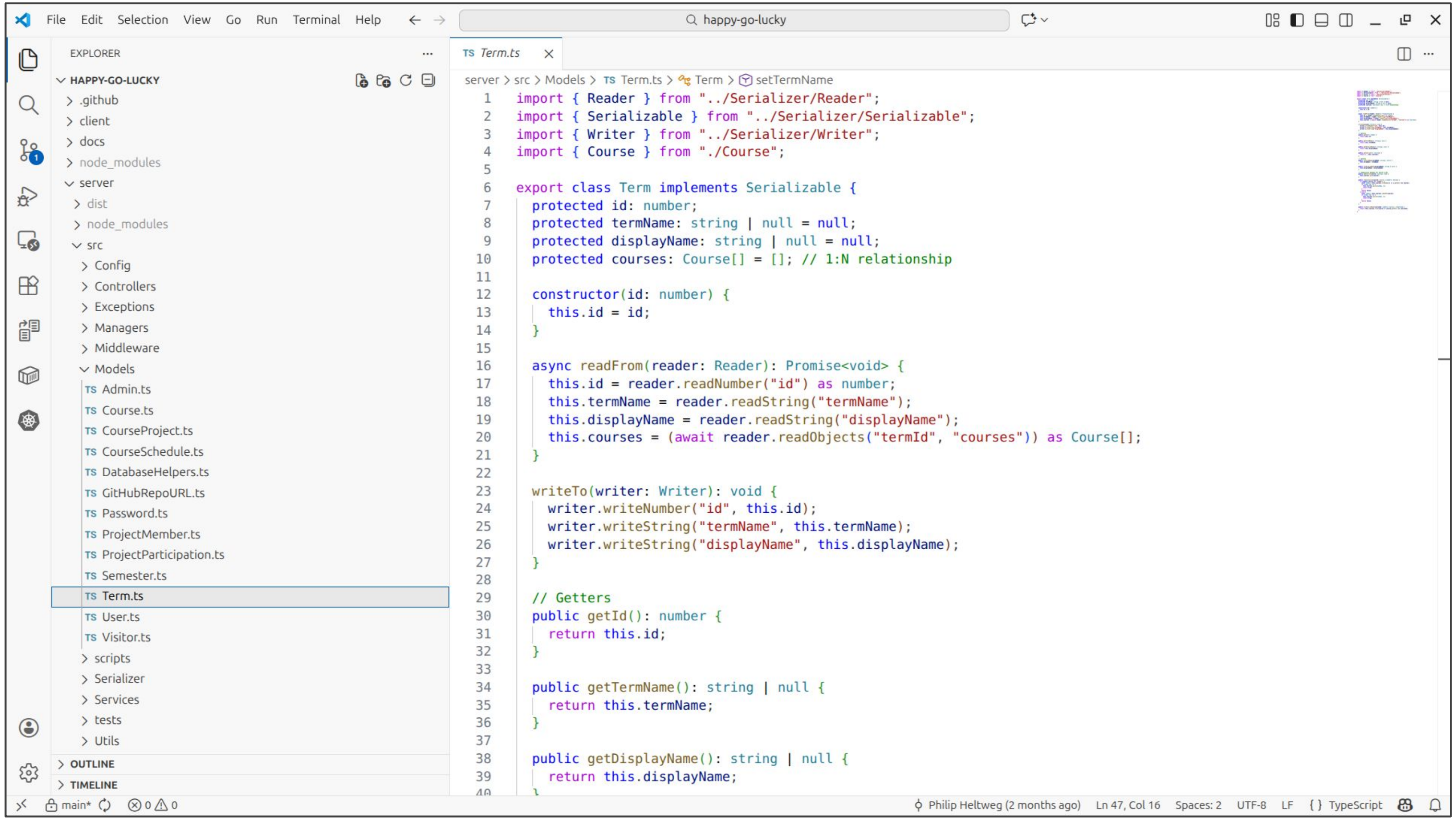
Original HGL Solutions

For a given project, a member

- Can enter their happiness
- Can send out stand-up emails
- Can review their coding activities

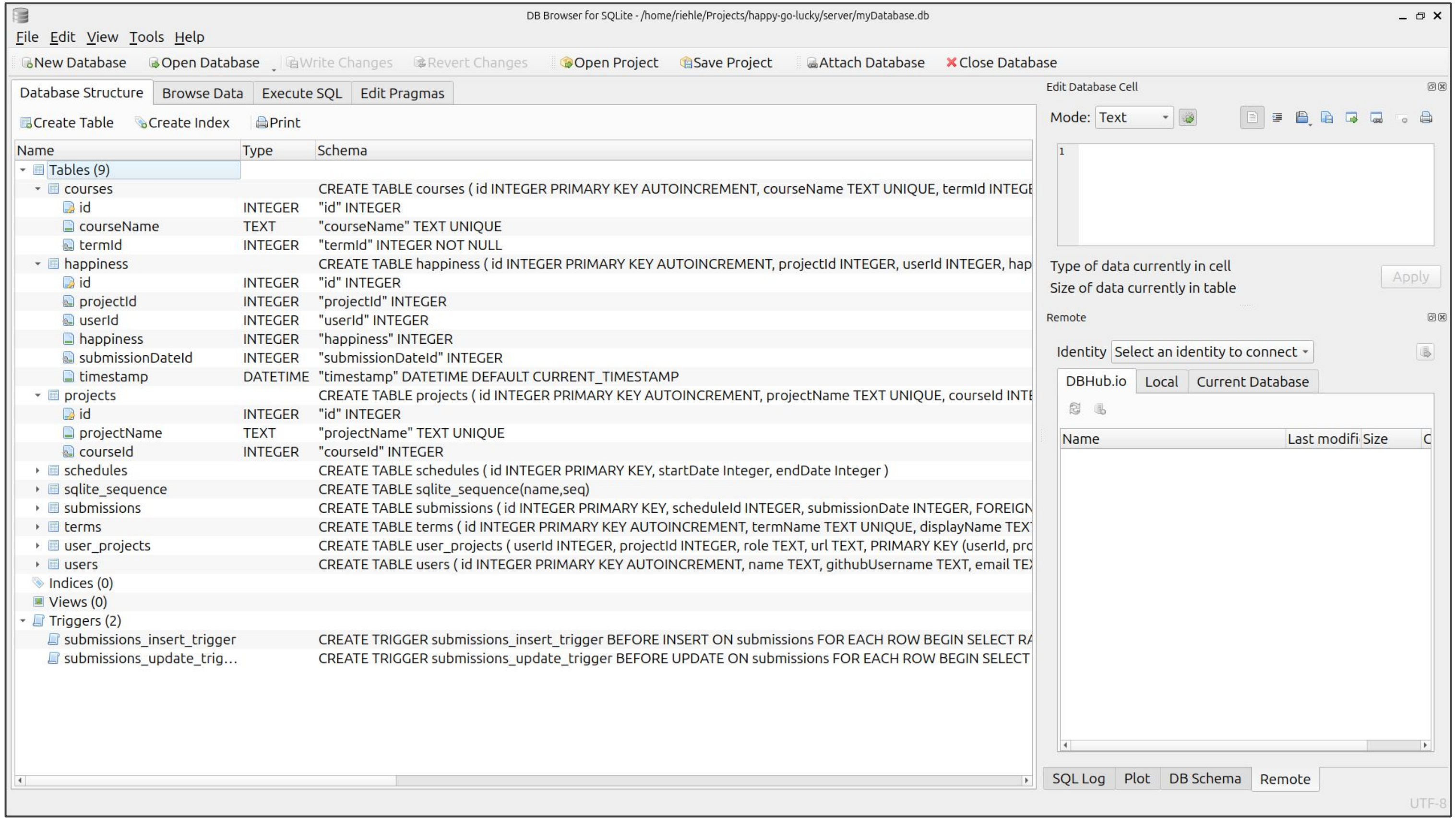
all scoped by the course schedule

Use Your Fav IDE But Keep Cruft Out



```typescript
import { Reader } from "../Serializer/Reader";
import { Serializable } from "../Serializer/Serializable";
import { Writer } from "../Serializer/Writer";
import { Course } from "./Course";

export class Term implements Serializable {
  protected id: number;
  protected termName: string | null = null;
  protected displayName: string | null = null;
  protected courses: Course[] = []; // 1:N relationship

  constructor(id: number) {
    this.id = id;
  }

  async readFrom(reader: Reader): Promise<void> {
    this.id = reader.readNumber("id") as number;
    this.termName = reader.readString("termName");
    this.displayName = reader.readString("displayName");
    this.courses = (await reader.readObjects("termId", "courses")) as Course[];
  }

  writeTo(writer: Writer): void {
    writer.writeNumber("id", this.id);
    writer.writeString("termName", this.termName);
    writer.writeString("displayName", this.displayName);
  }

  // Getters
  public getId(): number {
    return this.id;
  }

  public getTermName(): string | null {
    return this.termName;
  }

  public getDisplayName(): string | null {
    return this.displayName;
```

SQLite Database Browser (myDatabase.db)



Name	Type	Schema
Tables (9)		
courses		CREATE TABLE courses (id INTEGER PRIMARY KEY AUTOINCREMENT, courseName TEXT UNIQUE, termId INTEGE
id	INTEGER	"id" INTEGER
courseName	TEXT	"courseName" TEXT UNIQUE
termId	INTEGER	"termId" INTEGER NOT NULL
happiness		CREATE TABLE happiness (id INTEGER PRIMARY KEY AUTOINCREMENT, projectId INTEGER, userId INTEGER, hap
id	INTEGER	"id" INTEGER
projectId	INTEGER	"projectId" INTEGER
userId	INTEGER	"userId" INTEGER
happiness	INTEGER	"happiness" INTEGER
submissionDateId	INTEGER	"submissionDateId" INTEGER
timestamp	DATETIME	"timestamp" DATETIME DEFAULT CURRENT_TIMESTAMP
projects		CREATE TABLE projects (id INTEGER PRIMARY KEY AUTOINCREMENT, projectName TEXT UNIQUE, courseId INTE
id	INTEGER	"id" INTEGER
projectName	TEXT	"projectName" TEXT UNIQUE
courseId	INTEGER	"courseId" INTEGER
schedules		CREATE TABLE schedules (id INTEGER PRIMARY KEY, startDate Integer, endDate Integer)
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
submissions		CREATE TABLE submissions (id INTEGER PRIMARY KEY, scheduleId INTEGER, submissionDate INTEGER, FOREIGN
terms		CREATE TABLE terms (id INTEGER PRIMARY KEY AUTOINCREMENT, termName TEXT UNIQUE, displayName TEX
user_projects		CREATE TABLE user_projects (userId INTEGER, projectId INTEGER, role TEXT, url TEXT, PRIMARY KEY (userId, pro
users		CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, name TEXT, githubUsername TEXT, email TE
Indices (0)		
Views (0)		
Triggers (2)		
submissions_insert_trigger		CREATE TRIGGER submissions_insert_trigger BEFORE INSERT ON submissions FOR EACH ROW BEGIN SELECT RA
submissions_update_trig…		CREATE TRIGGER submissions_update_trigger BEFORE UPDATE ON submissions FOR EACH ROW BEGIN SELECT

Optional Homework

See this table (same as project, but use table)

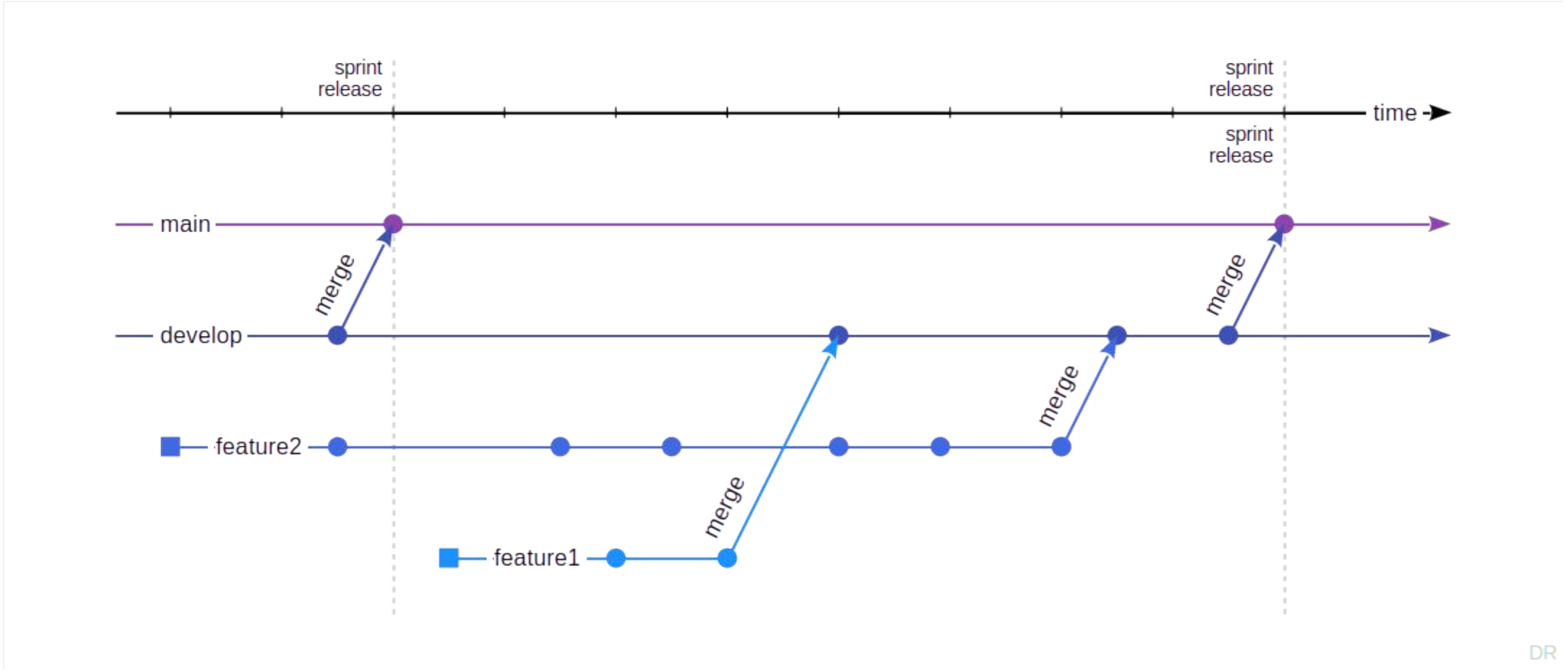
You will not be penalized if you don’t participate

You’ll get extra credit if you participate

Extra credit might lift your grade if you are at a grade boundary

View this experience as an idea of your first industry job

Working With Branches



Homework Work and Git Flow

Fork and work from your own repository

Pick and mark item to work on (spreadsheet)

Open pull request and explain what you are going to do

Semantically chunk your work, amend your pull request

Your work may or may not be integrated

Thank you! Any questions?

dirk.riehle@fau.de – https://oss.cs.fau.de

dirk@riehle.org – https://dirkriehle.com – @dirkriehle

Legal Notices