Конспект за Софтуерно инженерство

**29 седмици x 5 часа = 145 учебни часа**

## І. Общо представяне на учебната програма

Теория:

* Работен процес и agile development
* Системи за управление на проекти (примерно Trello, GitHub Issues)
* Софтуерно тестване
* Софтуерна документация

Практика:

* Практически проект с отворен код
  + Да се доработи съществуващ проект
  + Да се ползва управление на задачи, срокове
  + Да се напише документация
  + Да се напишат тестове

## ІІ. Цели на обучението по предмета

…

## ІІІ. Разпределение на учебното време

**29 седмици** x **5 часа** = **145 учебни часа**

## ІV. Учебно съдържание

|  |  |  |
| --- | --- | --- |
| **№** | **Наименование на разделите** | **Минимален брой часове** |
| 1. | **Въведение в курса**   * Въведение в софтуерната разработка (процеси, методологии, роли, инструменти) * Раздаване на практически проекти и разделяне по екипи * Дава се съществуващ проект, който всеки екип трябва да допише, да направи тестове, да надгради функционалност, да документира, да изгради CI система и да използва управление на процеси през цялото време | **5**  (2+3) |
| 2. | **Работа с чужд код**   * Работа с чужд проект * Разучаване на проекта, инсталация, конфигурация и стартиране на проекта * Дописване на нова функционалност към проекта * Упражнение: работа с чужд проект, инсталация и стартиране на съществуващ проект, дописване на функционалност | **5**  (2+3) |
| 3. | **Сорс-контрол системи**   * Сорс-контрол системи. Използване на Git и GitHub * Разлики между централизирана и децентрализирана сорс-контрол система * Упражнения: екипно взаимодействие с Git и GitHub, създаване на проекти, теглене и качване на промени, създаване и решаване на конфликти от конкурентни промени и сливане на конфликтни промени | **10**  (4+6) |
| 4. | **Софтуерни изисквания и прототипи**   * Анализ на софтуерните изисквания, случаи на употреба, истории (user stories), спецификация на изискванията (SRS), гъвкави изисквания * Протитипи на потребителския интерфейс (UI Prototyping) * Упражнение: създаване на UI прототип | **10**  (4+6) |
| 5. | **Софтуерно тестване**   * Писане на unit тестове и регресия * Подпъхване на функционалност (mocking) * Покритие на кода (code coverage) * Интеграционни тестове (integration testing) * Настройване на непрекъсната интеграция (GitHub + Travis CI) * Упражнение: писане на unit тестове и измерване на code coverage + mocking * Упражнение: писане на интеграционни тестове + fixture * Упражнение: имплементиране на непрекъсната интеграция | **25**  (10+15) |
| 6. | **Софтуерна документация**   * Писане на софтуерна документация (Readme, Wiki, ...) * Документиране на процес на инсталация и стартиране на проект и на частите на проект и процес на работа * Упражнение: документиране на код | **10**  (4+6) |
| 7. | **Процеси за софтуерна разработка**   * Основни методологии и техните характеристики * Гъвкави методологии: Scrum и Kanban * Запознаване с инструменти за управление на проекти и задачи (като Trello и GitHub Projects, GitHub Issues, ...) | **10**  (4+6) |
| 8. | **Екипна работа по проект**   * Екипна работа по практически проект, използвайки наученото в курса * Задължително се работи в екип и се използва сорс контрол система и тракер за задачи и дефекти | **60**  (24+36) |
| 9. | **Защита на екипен проект**   * Представяне, защита и оценяване на екипните проекти | **10**  (4+6) |
|  | **ОБЩО** | **145** |

V. Тематичен план

Като таблицата горе?

## VІ. Очаквани резултати от обучението

Покрива:

* EРУ 11. Работен процес
  + РУ 11.1. Владее основните процеси на софтуерно тестване
  + РУ 11.2. Използва методологии за разработка на софтуер
  + РУ 11.3. Създава софтуерна документация
  + РУ 11.4. Използва системи за контрол на версията на изходния код
  + РУ 11.5. Прилага процес на управление на задачите
  + РУ 11.6. Работи с чужд код

## VІІ. Авторски колектив

* Радослав Георгиев, Ивайло Бъчваров, <https://hackbulgaria.com> <https://hacksoft.io>

## VІІІ. Литература

1. Software Development, Design and Coding: With Patterns, Debugging, Unit Testing, and Refactoring 2nd edition, John F. Dooley, Apress, 2017, ISBN 978-1484231524
2. Refactoring: Improving the Design of Existing Code (2nd Edition), Martin Fowler, Addison-Wesley, 2018, ISBN 978-0134757599
3. The Pragmatic Programmer, Andy Hunt, Dave Thomas, Addison-Wesley, 1999, ISBN 978-0201616224
4. Pragmatic Thinking and Learning: Refactor Your Wetware (Pragmatic Programmers), Andy Hunt, Pragmatic Bookshelf, 2008, ISBN, 978-1934356050
5. Clean Code, Robert Martin, Prentice Hall, 2008, ISBN 978-0132350884
6. Code Complete: A Practical Handbook of Software Construction, Second Edition, Steve McConnell, Microsoft Press, 2004, 978-0735619678