Good Practices & Code Review

Alex Coll Latorre

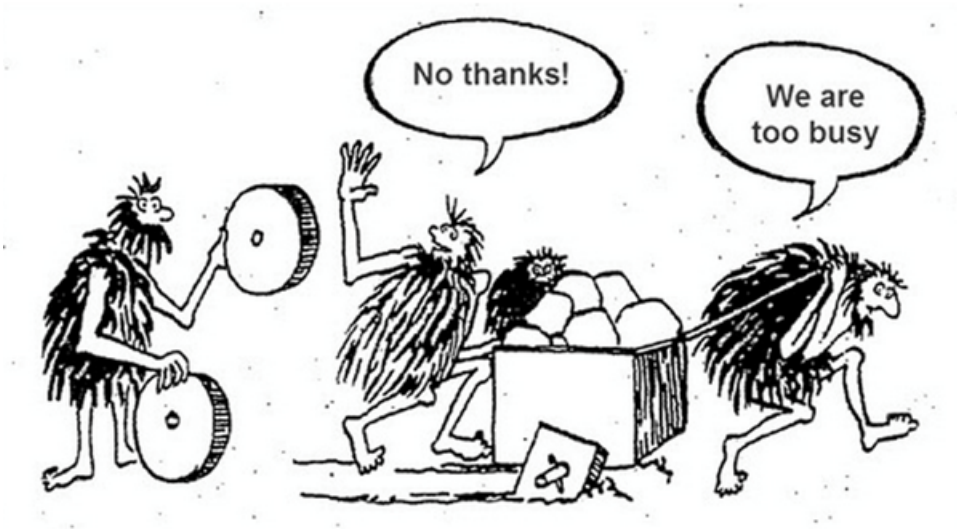
Diciembre 2021

Índice general

Capítulo 1

Deuda técnica

La deuda técnica es un concepto en el que se expone que el desarrollo de nuevas funcionalidades se ve ralentizado por la estructura previa del código[1].



[1] Más información en: https://martinfowler.com/bliki/TechnicalDebt.html y https://www.productplan.com/glossary/technical-debt/


TECHNICAL DEBT
I DON'T UNDERSTAND WHY IT TAKES SO LONG TO ADD A NEW WINDOW.
@VINCENTDNL

Capítulo 2

Monstruo

```
type GenericFormProps = {
  type: string //typeof Article | typeof Event | typeof Form | typeof
      Tag | typeof F
  data: any
  userID: string
  handlerSave: (formResult: any) => void
  handlerGoBack: () => void
  date?: Date
  parentIDFile?: string | null
}

const fields: { [key: string]: any } = {
  title: '',
  description: '',
  content: '',
  required: false,
  order: 0,
  category: ArticleCategory.Healthy,
  lang: LangType.Arab,
  files: [],
  metaTitle: '',
  metaDescription: '',
  ogTitle: '',
  ogDescription: '',
  ogImage: [],
  slug: '',
  tags: [],
  startDate: new Date(),
  finishDate: new Date(),
  repeating: '',
  isRepeatingUntil: false,
  repeatingUntil: new Date(),
  place: '',
  eventCategory: EventCategory.GeneralRevision,
  users: [],
```

```
  notifications: [],
  formFields: [],
  name: '',
  isSection: false,
  sectionSlug: '',
  parentSection: '',
  isCronShowed: false,
  isDir: false,
  resourceFile: [],
  isPublic: false,
  sharedWith: [],
}

export function GenericForm(props: GenericFormProps) {
  const { t } = useTranslation()

  const rowsInMultilineTextFields = 5

  const [fieldsState, setFieldsState] = useState(
    initializeFields(props.type, props.data, props.date)
  )
  const [fieldsMapped, setFieldsMapped] = useState<JSX.Element[]>()
  const [cleanFields, setCleanFields] = useState<boolean>(false)
  const [errorMessage, setErrorMessage] = useState<string>('')

  useEffect(() => {
    setFieldsMapped(createElementTags())
    setCleanFields(false)
  }, [fieldsState])

  function getAutocompleteIsMultiple(field: string): boolean {
    switch (field) {
      case PropertiesNames.USERS:
        return true
      case PropertiesNames.TAGS:
        return true
      case PropertiesNames.PARENT_SECTION:
        return false
      case PropertiesNames.SHARED_WITH:
        return true
      default:
        return false
    }
  }

  function isFieldMultiple(field: string): boolean {
    return (
      field.toLowerCase().includes(PropertiesNames.DESCRIPTION) ||
      field.toLowerCase().includes(PropertiesNames.CONTENT)
    )
  }

  function handleAnyChange(field: string, value: any) {
    const result: { [key: string]: any } = {}
    Object.getOwnPropertyNames(fieldsState).forEach((f: string) => {
      result[f] = field === f ? value : fieldsState[f]
    })
```

```
    setFieldsState(result)
  }

  function handleSave(event: React.FormEvent<HTMLFormElement>) {
    event.preventDefault()
    if (!validateForm()) return
    const result = createObject()
    if (!Object.keys(result).length) {
      return
    }
    let title = 'New'
    if (props.data) {
      title = 'Edited'
    }

    let description = ''
    let sourceType = SourceType.Calendar
    let url = ''
    if (result instanceof F) {
      description = result.name
      sourceType = SourceType.Files
      url = ROUTE_RESOURCES
    } else if (result instanceof Event) {
      description = result.title
      sourceType = SourceType.Calendar
      url = ROUTE_CALENDAR
    } else if (result instanceof Article) {
      description = result.title
      sourceType = SourceType.Content
      url = `${ROUTE_ARTICLES}/${result.id}`
    } else if (result instanceof Tag) {
      description = `${t('tagName')}: ${result.name}`
      sourceType = SourceType.Content
      url = ROUTE_TAG_LIST
    } else if (result instanceof Form) {
      description = `${t('formName')}: ${result.title}`
      sourceType = SourceType.Forms
      url = ROUTE_FORM_GENERATOR
    }

    const message = new Message({
      description,
      id: uuidv4(),
      message: '',
      notifications: [],
      sourceType,
      title: `${title} ${props.type}`,
      url,
    })
    const notification = new Notification({
      createdAt: new Date(),
      deliveryStatus: DeliveryStatus.Created,
      id: uuidv4(),
      message,
      priorityLevel: PriorityLevel.Normal,
      recipientID: props.userID,
      sendAt: new Date(),
```

```
      senderID: uuidv4(),
      transportType: TransportType.App,
    })
    message.notifications.push(notification)
    notificationService.add(notification)
    messageService.add(message)
    props.handlerSave(result)
  }

  function validateForm() {
    switch (props.type) {
      case TypesNames.FORM:
        if (fieldsState[PropertiesNames.FORM_FIELDS].length === 0) {
          setErrorMessage(ErrorMessages.FORM_FIELDS)
          return false
        }
        if (fieldsState[PropertiesNames.USERS].length === 0) {
          setErrorMessage(ErrorMessages.FORM_USERS)
          return false
        }
        break
      case TypesNames.ARTICLE:
        if (fieldsState[PropertiesNames.TAGS].length === 0) {
          setErrorMessage(ErrorMessages.TAGS)
          return false
        }
        break
      case TypesNames.RESOURCE:
        if (
          fieldsState[PropertiesNames.RESOURCE_FILE]?.length === 0 &&
          !fieldsState[PropertiesNames.IS_DIR]
        ) {
          setErrorMessage(ErrorMessages.RESOURCE_FILE)
          return false
        }
        break
      default:
        break
    }
    return true
  }

  return (
    <form onSubmit={handleSave} className={containerClassName(props.
        type)}>
      {fieldsMapped}
      <Box>
        {errorMessage ? (
          <Alert
            severity="warning"
            key="errorMessage"
            id="errorMessage"
            className={errorMessagesStyle(props.type)}>
            {t(errorMessage)}
          </Alert>
        ) : (
          ''
```

```
205           )}
206         </Box>
207         <Box key="buttons" className={styles.buttonsSpacingBetween}>
208           <Button variant="contained" color="secondary" onClick={props.
                handlerGoBack}>
209             {t('back')}
210           </Button>
211           <Button variant="contained" color="secondary" onClick={
                resetFields}>
212             {t('reset')}
213           </Button>
214           <Button variant="contained" color="primary" type="submit">
215             {t('save')}
216           </Button>
217         </Box>
218       </form>
219     )
220   }
221
222   function containerClassName(typeName: string) {
223     if (typeName === TypesNames.TAG) {
224       return styles.form + ' ' + styles.formSpacing
225     } else if (typeName === TypesNames.FORM) {
226       return styles.form + ' ' + styles.formSpacing
227     } else if (typeName === TypesNames.EVENT) {
228       return styles.errorMessageForm + ' ' + styles.form + ' ' + styles.
            formSpacing
229     } else if (typeName === TypesNames.ARTICLE) {
230       return styles.form + ' ' + styles.formSpacing
231     } else if (typeName === TypesNames.RESOURCE) {
232       return styles.form + ' ' + styles.formSpacing
233     }
234   }
235   function errorMessagesStyle(typeName: string) {
236     if (typeName === TypesNames.TAG) {
237       return styles.errorMessageTags
238     } else if (typeName === TypesNames.FORM) {
239       return styles.errorMessageForm
240     }
241   }
242
243   function initializeFields(prototype: any, data: any, date?: Date) {
244     let modelType = ''
245     switch (prototype) {
246       case TypesNames.FORM:
247         modelType = typeof Form
248         break
249       case TypesNames.TAG:
250         modelType = typeof Tag
251         break
252       case TypesNames.ARTICLE:
253         modelType = typeof Article
254         break
255       case TypesNames.EVENT:
256         modelType = typeof Event
257         break
258       case TypesNames.RESOURCE:
```

```
259         modelType = typeof F
260         break
261     }
262     const resultObject: { [key: string]: any } = {}
263     if (modelType === '') return resultObject
264     const propsFields = [
265       ...Object.getOwnPropertyNames(modelType),
266       ...getAdditionalFields(prototype.name),
267     ]
268     const elementFields = Object.keys(fields)
269     propsFields.forEach((field) => {
270       if (elementFields.includes(field)) {
271         Object.defineProperty(resultObject, field, {
272           value: data ? data[field] : fields[field],
273           writable: true,
274         })
275       }
276     })
277
278     //Needed to set dates in Event
279     if (resultObject[PropertiesNames.START_DATE]) {
280       resultObject[PropertiesNames.START_DATE] = getStartDate(date || new
              Date())
281     }
282
283     if (resultObject[PropertiesNames.FINISH_DATE]) {
284       resultObject[PropertiesNames.FINISH_DATE] = getFinishRepeatingDate(
              date || new Date())
285     }
286     if (resultObject[PropertiesNames.REPEATING_UNTIL]) {
287       resultObject[PropertiesNames.REPEATING_UNTIL] =
              getFinishRepeatingDate(date || new Date())
288     }
289
290     //Needed to set resource file in Resources
291     if (propsFields.includes(PropertiesNames.RESOURCE_FILE)) {
292       if (data) {
293         resultObject[PropertiesNames.RESOURCE_FILE] = []
294         resultObject[PropertiesNames.RESOURCE_FILE].push(data)
295       }
296     }
297
298     return resultObject
299   }
300
301   function getStartDate(date: Date) {
302     let startDate = new Date(date)
303     if (dateToDateString(date) === dateToDateString(new Date())) {
304       let newDate = new Date()
305       startDate.setHours(newDate.getHours(), newDate.getMinutes(),
              newDate.getSeconds())
306     } else {
307       startDate.setHours(0, 0, 0)
308     }
309
310     return startDate
311   }
```

```
312
313 function getFinishRepeatingDate(date: Date) {
314   let resultDate = new Date(date)
315   if (dateToDateString(date) === dateToDateString(new Date())) {
316     let newDate = new Date()
317     resultDate.setHours(newDate.getHours() + 1, newDate.getMinutes(),
          newDate.getSeconds())
318   } else {
319     resultDate.setHours(1, 0, 0)
320   }
321
322   return resultDate
323 }
```

Listing 2.1: El formulario no existe... Y menos mal 😕

Capítulo 3

Método LAAAAAAAAAAA...RGO

```
1  func (fh files) Create(ctx *routing.Context) error {
2      fv := vf.FileFront{}
3
4    if err := ctx.Read(&fv); err != nil {
5      return err
6    }
7
8    fileModel := fv.ToModelOwner(common.NewAppContext(ctx).UserID)
9    _, err := fh.fileSrv.ByID(common.NewAppContext(ctx), fileModel.ID())
10
11   if err != nil {
12     err := fh.fileSrv.Create(common.NewAppContext(ctx), fileModel)
13     if err != nil {
14       return routing.NewHTTPError(http.StatusInternalServerError, err.
             Error())
15     }
16
17     if fileModel.Parent() == common.FrontEndParentTypeArticle {
18       fileModel.SetParent("")
19     } else if fileModel.Parent() == common.FrontEndParentTypeEvent {
20       uidSystem := fh.fileSrv.GetSystem(common.NewAppContext(ctx),
             fileModel.Owner(), common.FolderCalendar)
21       if uidSystem != "" {
22         fileModel.SetParent(uidSystem)
23       }
24     } else {
25       uidSystem := fh.fileSrv.GetSystem(common.NewAppContext(ctx),
             fileModel.Owner(), fileModel.Parent())
26       if uidSystem != "" {
27         fileModel.SetParent(uidSystem)
28       }
29
30     }
31
32     if err := fh.fileSrv.UpdateSystem(common.NewAppContext(ctx),
```

```
          fileModel); err != nil {
33        return routing.NewHTTPError(http.StatusInternalServerError, err.
              Error())
34      }
35
36    } else {
37      err := fh.fileSrv.Update(common.NewAppContext(ctx), fileModel)
38      if err != nil {
39        return routing.NewHTTPError(http.StatusInternalServerError, err.
              Error())
40      }
41    }
42
43    if fv.MessageId != "" {
44      cm, err := fh.convSrv.ByID(common.NewAppContext(ctx), fv.MessageId)
45      if err != nil {
46        return routing.NewHTTPError(http.StatusInternalServerError, err.
              Error())
47      }
48
49      //usersConversation := make([]string, 0, len(cm.Users()))
50      for i := range cm.Users() {
51        if fileModel.Owner() != cm.Users()[i].UserID() {
52          //usersConversation = append(usersConversation, cm.Users()[i].
                UserID())
53          fs := models.NewSharedWith(cm.Users()[i].UserID(), fileModel.ID
                ())
54          err = fh.fileShared.CreateSharedWith(common.NewAppContext(ctx),
                 fs)
55
56          if err != nil {
57            return routing.NewHTTPError(http.StatusInternalServerError,
                  err.Error())
58          }
59
60          fileModel.SetShared(fs)
61        }
62      }
63    }
64
65    if len(fv.SharedWith) > 0 {
66      for i := range fv.SharedWith {
67        fs := models.NewSharedWith(fv.SharedWith[i], fileModel.ID())
68        err = fh.fileShared.CreateSharedWith(common.NewAppContext(ctx),
              fs)
69
70        if err != nil {
71          return routing.NewHTTPError(http.StatusInternalServerError, err
                .Error())
72        }
73
74        fileModel.SetShared(fs)
75      }
76    }
77
78    if !fileModel.IsDir() {
79      if _, err := os.Stat(fileUpload + fileModel.Owner()); os.IsNotExist
```

```
          (err) {
80        errDir := os.Mkdir(fileUpload+fileModel.Owner(), 0755)
81
82        if errDir != nil {
83          return routing.NewHTTPError(http.StatusInternalServerError,
                errDir.Error())
84        }
85      }
86
87      data, err := base64.StdEncoding.Strict().DecodeString(fv.Data)
88      if err != nil {
89        return routing.NewHTTPError(http.StatusInternalServerError, err.
              Error())
90      }
91
92      if err := ioutil.WriteFile(fileUpload+fileModel.Owner()+"/"+
            fileModel.ID()+"."+fileModel.Extension(), data, 0644); err !=
            nil {
93        return routing.NewHTTPError(http.StatusInternalServerError, err.
              Error())
94      }
95
96      fileSize := len(data)
97      fileModel.SetSize(fileSize)
98
99      if err := fh.fileSrv.UpdateSize(common.NewAppContext(ctx),
            fileModel); err != nil {
100       return routing.NewHTTPError(http.StatusInternalServerError, err.
              Error())
101     }
102   }
103
104   if err = ctx.Write(vf.FromModel(fileModel)); err != nil {
105     fh.log.Error(err.Error())
106   }
107
108   return nil
109 }
```

Listing 3.1: File

Capítulo 4

Círculos que dan vueltas

```
export function Circles() {

  useEffect(() => {
    const mapUserPathologies = new Map<User, String[]>()
    circles.items.forEach((c) => {
      pathologyService
        .getFilteredList(
          new Query({
            query: [new QueryParam<PathologyQuery>('userID', c.id)],
          })
        )
        .subscribe((res) => {
          mapUserPathologies.set(
            c,
            res.items.map((i) => i.name)
          )
          setUserPathology(mapUserPathologies)
        })
    })
  }, [circles])

  useEffect(() => {
    const mapUserPathologies = new Map<User, String[]>()
    allCircles.items.forEach((c) => {
      pathologyService
        .getFilteredList(
          new Query({
            query: [new QueryParam<PathologyQuery>('userID', c.id)],
          })
        )
        .subscribe((res) => {
          mapUserPathologies.set(
            c,
            res.items.map((i) => i.name)
          )
```

```
          setAllUserPathology(mapUserPathologies)
        })
    })
  }, [allCircles])

  useEffect(() => {
    const ids = loggedUserService.get()?.related.map((r) => r.id) || []
    ids.push(loggedUserService.get()?.id || '')

    userService
      .getFilteredList(
        new Query({
          pager,
          query: [
            {
              name: 'isCircle',
              value: new BoolQueryParam(true),
            },
            {
              name: 'ids',
              value: ids,
            },
            ...searcherQuery(searcher),
          ],
        })
      )
      .subscribe((cs) => {
        setAllCircles(cs)
      })
  }, [])

  useEffect(() => {
    const ids = loggedUserService.get()?.related.map((r) => r.id) || []
    ids.push(loggedUserService.get()?.id || '')
    userService
      .getFilteredList(
        new Query({
          pager,
          query: [
            {
              name: 'isCircle',
              value: new BoolQueryParam(true),
            },
            {
              name: 'ids',
              value: ids,
            },
            ...searcherQuery(searcher),
          ],
        })
      )
      .subscribe((cs) => {
        setCircles(cs)
        if (cs.count === 1) {
          handleSelect(cs.items[0])
        }
      })
```

```
  }, [isLoading])

  useEffect(() => {
    const auxPath = emptyList<String>()
    pathologyService.getFilteredList(new Query({})).subscribe((res) =>
        {
      auxPath.items = res.items.map((i) => i.name)
      auxPath.count = res.count
      setPathologies(auxPath)
    })
  }, [])

  useEffect(() => {
    const ids = loggedUserService.get()?.related.map((r) => r.id) || []
    ids.push(loggedUserService.get()?.id || '')
    userService
      .getFilteredList(
        new Query({
          pager,
          query: [
            {
              name: 'isCircle',
              value: new BoolQueryParam(true),
            },
            {
              name: 'ids',
              value: ids,
            },
            ...searcherQuery(searcher),
          ],
        })
      )
      .subscribe((cs) => {
        setCircles(cs)
      })
  }, [pager, searcher])

  const action: Action = {
    handleAction: (
      event: ChangeEvent<{}>,
      value: String | null,
      reason: AutocompleteChangeReason,
      details?: AutocompleteChangeDetails<String> | undefined
    ) => {
      if (value) {
        if (filtering) {
          const us = emptyList<User>()
          circles.items.forEach((c) => {
            if (userPathology.get(c)?.includes(value)) {
              us.items.push(c)
            }
            us.count = us.items.length
            setCircles(us)
          })
        } else {
          const us2 = emptyList<User>()
          allCircles.items.forEach((c) => {
```

```
            if (allUserPathology.get(c)?.includes(value)) {
              us2.items.push(c)
            }
            us2.count = us2.items.length
            setCircles(us2)
          })
        }
      } else {
        if (!isLoading) {
          setIsLoading(true)
        } else {
          setIsLoading(false)
        }
      }
    },
  }

  const search: Search<UserQuery> = {
    searchValues: searcher,
    handleSearch: (svs: SearchValue<UserQuery>[]) => {
      const result: SearchValue<UserQuery>[] = []

      if (svs[0].value === undefined) {
        setFiltering(false)
      } else {
        setFiltering(true)
      }

      svs.forEach((s) => {
        if (s.type === 'date' && s.value) {
          result.push({
            name: s.name,
            type: s.type,
            label: s.label,
            value: new Date(s.value as string).toISOString().substring
                (0, 10),
          })
        } else {
          result.push(s)

          let value
          if (s.value) {
            value = s.value.charAt(0).toUpperCase() + s.value.slice(1)
          }

          result.push({
            name: s.name,
            type: s.type,
            label: s.label,
            value: value,
          })
        }
      })

      setSearcher(result)
    },
  }
```

```
205
206 }
```

Listing 4.1: Círculos

Capítulo 5

Buenas prácticas

5.1. Declaración de componentes

Más información en: https://www.twilio.com/blog/react-choose-functional-components

```
1  // 1st: Functional component vs Class component
2  export const Component = (props: ComponentProps) => {
3    // 2nd: State variables: typing, initial State
4    const [state, setState] = useState<boolean>(false)
5    const [name, setName] = useState<string>('')
6
7    // 3rd: useEffect in priority order: infinite (no dependencies),
         initial ("[]"), and others
8    useEffect(() => {
9      // In each render
10   })
11
12   useEffect(() => {
13     // Initial render
14   }, [])
15
16   // 4th: Correctly named functions with the following form:
17   const changeName = (newName: string) => setName(newName)
18   // Note: if you can avoid "{}", do it
19
20   const getState = (): string => '' + state
21
22   // 5th: JSX.Elements
23   return (
24     <>
25         {// Note: avoid "?", unless it's necessary. Use "&&" instead}
26         {state && <TextField/>}
27         {/* With "?", it would be:
28         {state ? (<TextField/>):null}
29         */}
30     </>
```

```
  )

  // 6th, but not least: DO NOT write code (or comments) in spanish :')
}
```

Listing 5.1: Convenciones para React

```
import styles from './Component.module.css';

...

const useStyles = makeStyles((theme) => ({
  box: {
    backgroundColor: 'yellow',
  },
}

const classes = useStyles()

...

const MyTitle = styled.div`
    background: yellow
`

return (
    <>
        {// 1st: Inline CSS (through style or className}
        <Box style={{ backgroundColor: 'yellow' }}></Box>
        <Box className={classes.box}></Box>

        {// 2nd: CSS module}
        <Box className={styles.button}></Box>

        {// 3rd: CSS in JS}
        <MyTitle>My First CSS-in-JS React component!</MyTitle>
    </>
)
```

Listing 5.2: Estilos

```
const fields: Field<Test>[] = [
    // Task: I want to show the number of a test's algorithm version,
        but it can be undefined or null
    {
      sortable: true,
      label: t('version'),
      name: 'algorithmVersion',
      renderFunc: (f, i) => i.algorithmVersion || '',
    },

    // But, what if we are in the "0" version?

    {
```

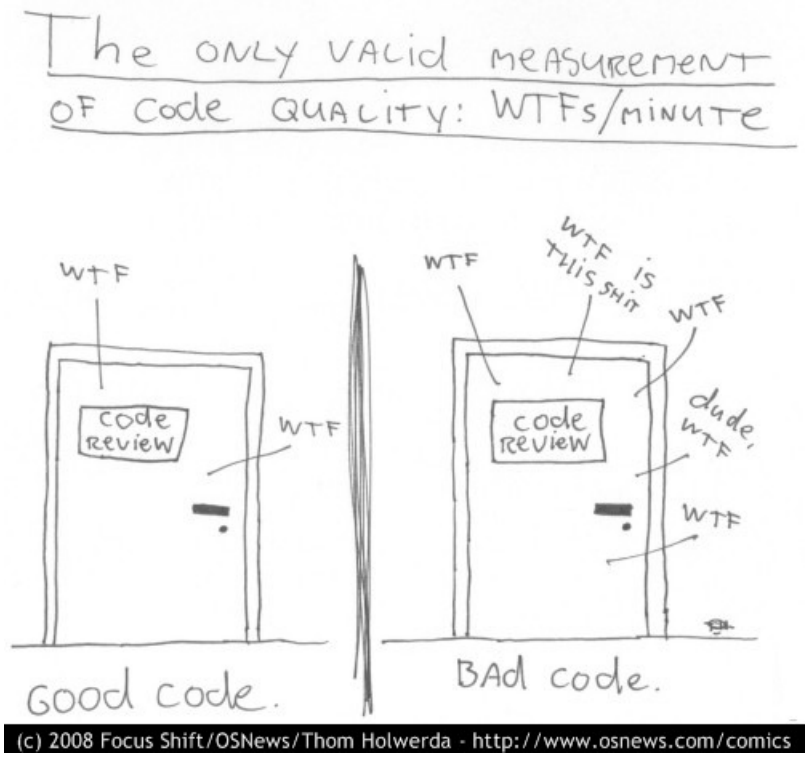
```
13        ...
14        renderFunc: (f, i) => i.algorithmVersion !== undefined && i.
              algoritmhVersion !== null ? i.algorithmVersion ? '',
15      },
16
17      // Best way: using nullish operator: ??
18      {
19        ...
20        renderFunc: (f, i) => '' + (i.algorithmVersion ?? ''),
21      },
22  ]
```

Listing 5.3: JavaScript

Capítulo 6

Recomendaciones

- Fowler, M., Beck, K., Brant, J., Opdyke, W., & Roberts, D. (1999). Refactoring: Improving the Design of Existing Code (1.a ed.). Addison-Wesley Professional.
- Freeman, E., Freeman, E., Sierra, K., Bates, B., & Robson, E. (2004). Head First Design Patterns. Van Duuren Media.
- Gamma, E., Helm, R., Dr, J. R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software (1.a ed.). Addison-Wesley Professional.
- Kerievsky, J. (2005). Refactoring to Patterns. Addison-Wesley.
- Krug, S., Bayle, E., & Matcho, M. (2014). Don't Make Me Think, Revisited. New Riders.
- Martin, R. (2008). Clean Code: A Handbook of Agile Software Craftsmanship;Robert C. Martin (Illustrated ed.). Pearson.
- https://javascript.info/
- https://www.codeinwp.com/blog/react-best-practices/
- https://betterprogramming.pub/21-best-practices-for-a-clean-react-project-df788a682fb
- https://deepsource.io/blog/javascript-code-quality-best-practices/


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift/OSNews/Thom Holwerda - http://www.osnews.com/comics